

System 800xA Information Management

Display Services

System Version 6.0

NOTICE

This document contains information about one or more ABB products and may include a description of or a reference to one or more standards that may be generally relevant to the ABB products. The presence of any such description of a standard or reference to a standard is not a representation that all of the ABB products referenced in this document support all of the features of the described or referenced standard. In order to determine the specific features supported by a particular ABB product, the reader should consult the product specifications for the particular ABB product.

ABB may have one or more patents or pending patent applications protecting the intellectual property in the ABB products described in this document.

The information in this document is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this document.

Products described or referenced in this document are designed to be connected, and to communicate information and data via a secure network. It is the sole responsibility of the system/product owner to provide and continuously ensure a secure connection between the product and the system network and/or any other networks that may be connected.

The system/product owners must establish and maintain appropriate measures, including, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of antivirus programs, and so on, to protect the system, its products and networks, against security breaches, unauthorized access, interference, intrusion, leakage, and/or theft of data or information.

ABB verifies the function of released products and updates. However system/product owners are ultimately responsible to ensure that any system update (including but not limited to code changes, configuration file changes, third-party software updates or patches, hardware change out, and so on) is compatible with the security measures implemented. The system/product owners must verify that the system and associated products function as expected in the environment they are deployed.

In no event shall ABB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from ABB, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license. This product meets the requirements specified in EMC Directive 2004/108/EC and in Low Voltage Directive 2006/95/EC.

TRADEMARKS

All rights to copyrights, registered trademarks, and trademarks reside with their respective owners.

Copyright © 2003-2016 by ABB.
All rights reserved.

Release: September 2016
Document number: 3BUF001093-600 A

Table of Contents

About This User Manual

| | |
|--|----|
| General | 13 |
| User Manual Conventions | 15 |
| Warning, Caution, Information, and Tip Icons | 15 |
| Intended User..... | 16 |
| Terminology..... | 16 |
| Released User Manuals and Release Notes | 16 |

Section 1 - Introduction

| | |
|--------------------------------------|----|
| Prerequisites and Requirements | 20 |
|--------------------------------------|----|

Section 2 - Setup and Startup

| | |
|--|----|
| Startup | 21 |
| Setting Up PC Client Environments..... | 21 |
| Application Start-up | 23 |
| Checking Display Server Status..... | 27 |
| Stopping the Display Server..... | 29 |
| Launching the DisplayIE Client as an ActiveX Control | 30 |

Section 3 - A Quick Tutorial

| | |
|---|----|
| Introduction | 35 |
| Building a Basic Display for Process Data..... | 35 |
| How to Create a New Display | 36 |
| How to Insert a Display Element | 38 |
| How to Configure Display Element Properties | 40 |
| How to Define a Query | 43 |

| | |
|---|----|
| Saving Your Work | 44 |
| How to Test Your Display | 45 |
| How to Leave the Run Mode and Exit Display Services | 46 |
| Building a Trend Display..... | 46 |

Section 4 - User Interface & Application Tips

| | |
|--|----|
| Introduction | 57 |
| User Interface | 57 |
| Mouse Operation | 58 |
| Object Browser..... | 58 |
| Toolbar | 61 |
| Main Menu Bar | 62 |
| Application Guidelines | 68 |
| Performance | 68 |
| Application Hints | 68 |
| SQL Queries | 69 |
| SQL Queries for Numeric Log Data | 70 |
| Display Applications Index..... | 70 |

Section 5 - Display Services Configuration

| | |
|-------------------------------------|-----|
| Introduction | 73 |
| Working with Displays | 73 |
| Properties Dialog..... | 75 |
| Display Properties | 87 |
| Using a Grid | 91 |
| Aligning Display Elements | 92 |
| Distributing Display Elements | 94 |
| Grouping Display Elements | 94 |
| Ordering Display Elements | 95 |
| Element List | 97 |
| Working with Display Elements..... | 102 |
| Timer | 103 |
| Polyline | 105 |

| | |
|--|-----|
| Polygon | 106 |
| Shape | 108 |
| Text | 112 |
| Numeric | 115 |
| MultiText | 122 |
| List | 126 |
| Matrix | 131 |
| Edit | 145 |
| Combobox | 149 |
| Button | 152 |
| ABBBButton | 155 |
| Bar | 157 |
| Multibar | 162 |
| Pie | 167 |
| Trend | 170 |
| Gauge | 197 |
| SmartShade | 201 |
| XYPlot | 209 |
| ActiveX Control | 216 |
| Basic Properties | 218 |
| Working with User Elements | 227 |
| Encapsulation | 229 |
| Parent & This | 231 |
| User Element Tutorial | 232 |
| Properties for the User Element Definition | 249 |
| User Properties Dialog | 252 |
| Create New Property Dialog | 253 |
| Modify Property Dialog | 255 |
| Properties for User Element Instances | 259 |
| Protecting Tool | 266 |
| Trace Log | 268 |
| Communication Statistics | 269 |

| | |
|---|-----|
| Run-time Considerations | 270 |
| How to Open a Display in Run Mode | 270 |
| Runtime Diagnostics | 270 |
| PC-Client | 271 |
| Color Mapping for Printing..... | 272 |

Section 6 - Display Scripting

| | |
|--|-----|
| Introduction | 277 |
| Section Reference | 277 |
| Conventions | 278 |
| General Syntax for Script Statements..... | 278 |
| Data Types, Operators, and Expressions | 278 |
| Variable Names | 279 |
| Data Types..... | 280 |
| Constants | 280 |
| Arithmetic Operators..... | 281 |
| Relational and logical Operators | 282 |
| Bitwise Operators..... | 282 |
| Type Conversions | 283 |
| Comments | 283 |
| Control Flow Statements | 283 |
| If-Then-Else-Endif | 283 |
| Do-While | 284 |
| Queries and Actions | 285 |
| Queries | 285 |
| ReQueries..... | 285 |
| Actions | 286 |
| Functions | 286 |
| System variable | 286 |
| Session variable..... | 287 |
| Query | 287 |
| Action | 288 |
| String | 288 |

| | |
|-----------------------|-----|
| Math | 289 |
| General Purpose | 290 |
| acos | 290 |
| ascii | 291 |
| asin | 291 |
| asql | 291 |
| atan | 292 |
| basedisplay | 292 |
| bit | 292 |
| broadcast | 293 |
| chr | 293 |
| clientinfo | 293 |
| columns | 294 |
| cos | 294 |
| close | 294 |
| data | 295 |
| dcsgetobjinfo | 300 |
| dcssub | 300 |
| dec | 302 |
| delete | 302 |
| dialog | 303 |
| display | 303 |
| displayexist..... | 304 |
| dlovar | 304 |
| dsevar | 305 |
| execute | 305 |
| exit | 306 |
| file | 306 |
| getenv | 307 |
| getpref | 307 |
| group | 307 |
| hex | 308 |

| | |
|-------------|-----|
| iif | 308 |
| insert | 308 |
| instr | 309 |
| isevar | 309 |
| ishost | 310 |
| length | 310 |
| lower | 310 |
| lpad | 311 |
| lsyvar | 311 |
| ltrim | 311 |
| opendisplay | 312 |
| oscolor | 312 |
| osfont | 312 |
| ostreatment | 313 |
| print | 313 |
| rand | 314 |
| rearrange | 314 |
| replace | 315 |
| return | 315 |
| rpadd | 315 |
| rtrim | 316 |
| setfocus | 316 |
| sin | 317 |
| sql | 317 |
| sqrt | 318 |
| substr | 318 |
| system | 319 |
| tan | 319 |
| time | 320 |
| trace | 323 |
| update | 323 |
| upper | 324 |

| | |
|---|-----|
| Introduction | 327 |
| Tutorial 1, Hello World! | 335 |
| Tutorial 2, An Object Display | 337 |
| Introduction | 339 |
| How to Proceed | 339 |
| Accessing the Log Manager Trend Display | 340 |
| Configuration..... | 341 |
| Customizing the Display | 343 |
| Trace Area..... | 345 |
| Ruler | 345 |
| Trend Info..... | 345 |
| Scope Keys | 347 |
| Ruler Step Keys..... | 347 |
| Menu Bar | 347 |
| Introduction | 353 |
| Display Directory | 355 |
| Job Summary | 357 |
| Batch Summary | 360 |
| Batch Record Variables | 363 |
| Batch Trend | 367 |
| Lab Data Entry | 370 |
| Revision History | |
| Introduction | 373 |
| Revision History | 373 |
| Updates in Revision Index A..... | 373 |

Index

About This User Manual

General



Any security measures described in this User Manual, for example, for user access, password security, network security, firewalls, virus protection, etc., represent possible steps that a user of an 800xA System may want to consider based on a risk assessment for a particular application and installation. This risk assessment, as well as the proper implementation, configuration, installation, operation, administration, and maintenance of all relevant security related equipment, software, and procedures, are the responsibility of the user of the 800xA System.

This User Manual provides instructions for configuring and building custom graphic displays using Display Services for the 800xA System.

For instructions on configuring other applications, including data access, Softpoint Services, Calculations, and History Services, refer to *System 800xA Information Management Configuration (3BUF001092*)*.

The following are some quick guidelines to help you find what you are looking for in this book.

- Getting Acquainted with Display Services Functionality

Before you actually begin building displays, you may want to refer to [Appendix A, A Demonstration of Display Services](#). This is a brief guided tour of some demonstration displays provided with Display Services. This tour shows you some common applications for Display Services, and how to use display elements.

You can browse these displays and find examples of individual display elements, or even whole displays that you can use as blueprints for your own displays. In some cases you may only have to change tag references to use the pre-defined display elements.

- Learning the Basics

To start Display Services, see [Startup](#) on page 21. [Section 3, A Quick Tutorial](#) shows you how to create and test a display by inserting and configuring display elements. This tutorial takes about 15 minutes to complete. The tutorial does not demonstrate every single function available in Display Services. Still, if you have mastered the functions that are shown, you should be able to apply those skills to perform any Display Services function.

- Set-up

To set up language and user preferences refer to *System 800xA Information Management Configuration (3BUF001092*)*. Other set-up procedures are covered in [Section 2, Setup and Startup](#).

- Performance Considerations and Application Tips

Read [Section 4, User Interface & Application Tips](#) to get acquainted with the user interface, and for performance guidelines and application tips. This will help you avoid time-consuming errors and other problems.

- Application Index

[Display Applications Index](#) on page 70 provides a listing of specific display applications, with quick guidelines on how to implement them.

- Adding Displays

To create a new display and configure display properties, see [Working with Displays](#) on page 73.

- Adding, Elements

To add display elements to the display, and configure display element properties, see [Working with Display Elements](#) on page 102.

- Adding Symbol Library

As an option, you can make a library of custom display symbols by combining standard display elements. See [Working with User Elements](#) on page 227.

User Manual Conventions

Microsoft Windows conventions are normally used for the standard presentation of material when entering text, key sequences, prompts, messages, menu items, screen elements, etc.

Warning, Caution, Information, and Tip Icons

This User Manual includes Warning, Caution, and Information where appropriate to point out safety related or other important information. It also includes Tip to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:



Electrical warning icon indicates the presence of a hazard that could result in *electrical shock*.



Warning icon indicates the presence of a hazard that could result in *personal injury*.



Caution icon indicates important information or warning related to the concept discussed in the text. It might indicate the presence of a hazard that could result in *corruption of software or damage to equipment/property*.



Information icon alerts the reader to pertinent facts and conditions.



Tip icon indicates advice on, for example, how to design your project or how to use a certain function

Although Warning hazards are related to personal injury, and Caution hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, fully comply with all Warning and Caution notices.

Intended User

As a prerequisite you must know how to operate and navigate in the Windows platform where you are running Display Services, and you must be able to write queries with ORACLE SQL*Plus.

This user manual is intended for application engineers who are responsible for configuring and maintaining the custom graphic displays for data access for information management applications. This user manual is not the sole source of instruction for this functionality. It is recommended that you attend the applicable training courses offered by ABB.

Terminology

A complete and comprehensive list of terms is included in *System 800xA System Guide Functional Description (3BSE038018*)*. The listing includes terms and definitions that apply to the 800xA System where the usage is different from commonly accepted industry standard definitions and definitions given in standard dictionaries such as Webster's Dictionary of Computer Terms. Terms that uniquely apply to this User Manual are listed in the following table.

Released User Manuals and Release Notes

A complete list of all User Manuals and Release Notes applicable to System 800xA is provided in *System 800xA Released User Documents (3BUA000263*)*.

System 800xA Released User Documents (3BUA000263)* is updated each time a document is updated or a new document is released. It is in pdf format and is provided in the following ways:

- Included on the documentation media provided with the system and published to ABB SolutionsBank when released as part of a major or minor release, Service Pack, Feature Pack, or System Revision.
- Published to ABB SolutionsBank when a User Manual or Release Note is updated in between any of the release cycles listed in the first bullet.



A product bulletin is published each time *System 800xA Released User Documents (3BUA000263*)* is updated and published to ABB SolutionsBank.

Section 1 Introduction

Display Services Overview

This overview provides a quick summary of Display Services. Display Services let you build and view dynamic runtime displays on client nodes connected to a Display server. This is an efficient and cost-effective method for building and distributing process monitoring capability in your plant.

You can connect your custom-built displays to a variety of data sources. For example, you can use Display Services to build and run a display for viewing trend data collected by Information Manager History Services. You may also connect the same display to a third-party database, for instance an Oracle database on a remote Information Manager node.

Display Server

The Display Server is a collection of service and data providers for display call-up, management, and data access. The Service Provider manages licensing, user authority, and user preferences. In addition, the Service Provider supports the interface between the data providers (described below) and Display Clients.

Data providers let you connect your displays to historical and real-time process data, softpoints, alarm/event messages, and production data. The Display Data Provider manages display generation when a request for display is issued.

These data providers are described in greater detail in the section on configuring data access in *System 800xA Information Management Configuration* (3BUF001092*).

Display Client

The Display Client provides the graphical user interface for display building and viewing, and for administration of Display Services. You can connect up to 32

display clients (MDI plus SDI) and 32 DataDirect clients to a display server, [Figure 1](#).

You can also configure an external application such as Visual Basic or Internet Explorer to launch the Display Client as an ActiveX control. See [Launching the DisplayIE Client as an ActiveX Control](#) on page 30.

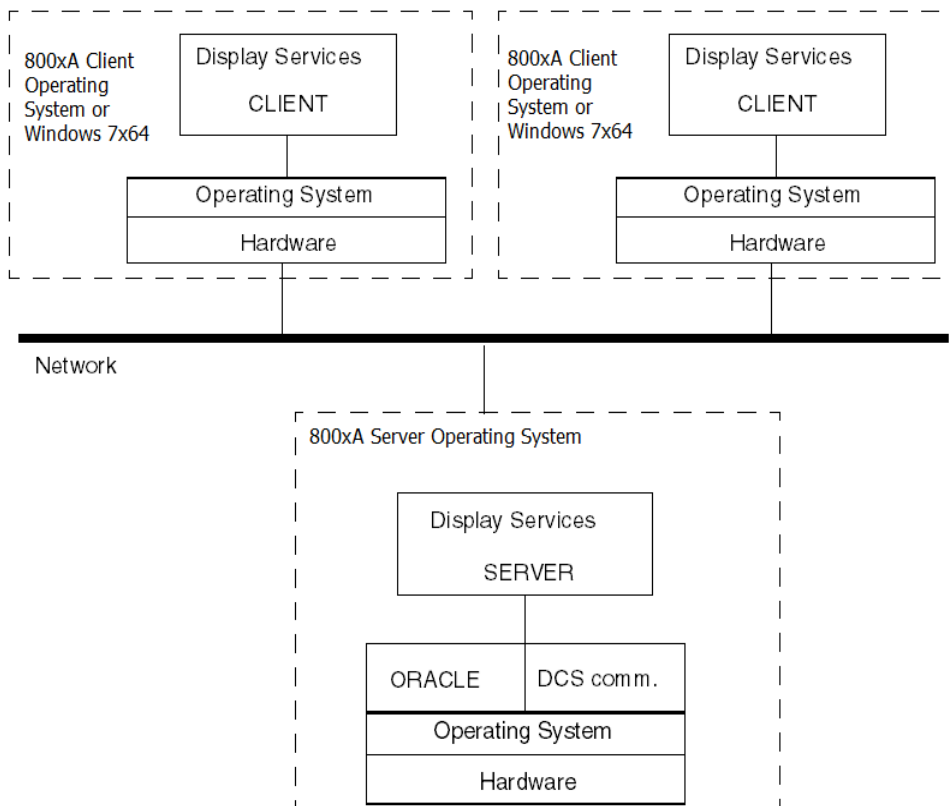


Figure 1. Display Services Architecture

Supported Platforms

The client and server software components can be loaded on 64-bit (x64) platform:

- Display Server - All supported Windows Server and Windows Client versions for SV6.0 800xA Nodes. See *Third Party Software System 800xA (3BUA000500)* document for all third party supported software.
- Display Client - In addition to supporting all OS versions supported by Display Server, Windows 7 x64 Professional, Ultimate, or Enterprise is also supported for Display Client. You must have at least one PC client for display building.

Display Services has two modes of operation:

- The Build mode is used to build dynamic runtime displays. This involves drawing display elements using typical drawing tools, and then configuring the properties of these display elements.

Display element properties are configured via an interactive dialog. Some dynamic properties such as data queries and actions are specified through scripts. The window where you define display element properties has an area for writing scripts.

Keywords, statements, functions, and data types for scripts are described in detail in [Section 6, Display Scripting](#).

- The Run mode is used to execute the displays in runtime to view runtime data.

Data providers can be installed on remote servers. This lets you access data from multiple sites (worldwide) for presentation on the same display. A maximum of 25 data providers can be connected to a server.

Additional Build and Data Provider Licenses

Additional build licenses are now available. The basic unit includes six data providers:

- one for real-time data access from the Advant OCS.
- one for historical data access from the Advant OCS.
- one for OPC data access.
- one for Oracle data access.
- one to support the demonstration displays described in [Appendix A, A Demonstration of Display Services](#).

- one for display definition.

Additional data providers can be licensed to provide additional data channels connecting to the server. These can be located on remote stations. For details on configuring data access, see the section on configuring data access in *System 800xA Information Management Configuration (3BUF001092*)*.

Prerequisites and Requirements

Server and Client Combined

Both the Display Services server and client functions are installed on the Information Manager server. The prerequisite hardware and software for the Information Manager server is located in *System 800xA Manual Installation (3BSE034678*)*.

PC-based Client

The Display Client software may also be installed on remote PC clients. Any graphic resolution (VGA, SVGA, and so on) can be used; however, the higher the resolution the better. The minimum requirements are:

- Windows 7 professional or higher, x64:
 - 256-color palette
 - 1 GHz or faster 64-bit (x64) processor
 - 2 GB RAM
 - 32 GB available hard disk space

The Display Services Open Data Access ODBC Client and Server option installed on a PC-based client occupies about 3 MB of hard disk space.

SQL Access to Numeric Log Data

Open Data Access ODBC software is installed with Information Management - History Services. SQL access to the oracle instance provides access to all the Open Data Access tables, including "NUMERICLOG".

Section 2 Setup and Startup

Startup



Display Services uses help files in Win Help (Winhlp32) format. Microsoft has discontinued support for this format. In order to view help files, download a viewer available from reference Microsoft Article ID: 917607 for more details on how to view winhelp32 format files.

This section describes the following application setup and start-up procedures:

- [Setting Up PC Client Environments](#) on page 21
- [Application Start-up](#) on page 23
- [Launching the DisplayIE Client as an ActiveX Control](#) on page 30

Setting Up PC Client Environments

Environment variables for clients include directory specifications for display caching (PC clients only), bitmap files, and log files, as well as sizing for user element caching, and communication time-out interval. If you need to modify any of these environment variables, use the Client Settings dialog as described below.

From the task bar, choose: **ABB Start Menu>ABB Industrial IT 800xA>Information Mgmt>Display Services>Client>IM Display Client Settings**. This displays the Client Settings dialog, [Figure 2](#).

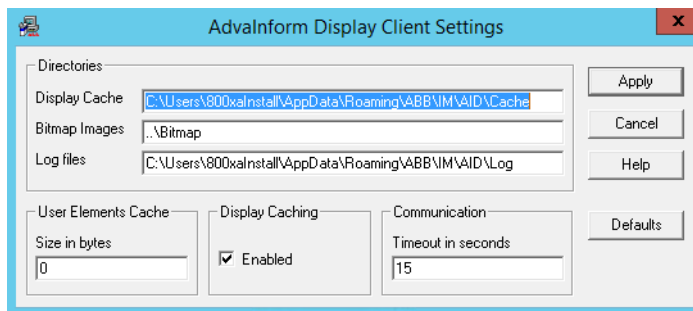


Figure 2. Client Settings Dialog

The following PC environment parameters may be configured:

[Display Cache Directory](#)

[Bitmap Images Directory](#)

[Log Files Directory](#)

[User Elements Cache, Size in Bytes](#)

[Display Caching Enable/Disable](#)

[Communication, Timeout in Seconds](#)

Display Cache Directory

A cache directory is created for each server with which a client connects. If display caching is enabled (see [Display Caching Enable/Disable](#) on page 22), this variable specifies the path to the directory for storing displays locally. Display caching speeds up the display loading time because each display is only loaded over the network the first time it is accessed (or if it is modified). If display caching is enabled, when a display is loaded the Client checks if the specified display exists locally. If it does, the client then requests the date of the display from the server. If the date is newer, the display is loaded from the server.

Display Caching Enable/Disable

This enables/disables display caching locally in the [Display Cache Directory](#). It is recommended that display caching be enabled for user elements for better performance.

User Elements Cache, Size in Bytes

This is the size of the User Element cache buffer. User Elements can be cached in memory for faster display loading (unlike displays which are cached in files). The Cache Size parameter determines the size of memory (in bytes) for caching User Elements.



Caching User Elements does not involve date comparison. Once a User Element is cached in memory, it will be used for all following instances even if a newer version exists. Therefore, during development and testing of User Elements, the Cache Size should be set to 0 (OFF), to ensure that you always use the latest version.

Bitmap Images Directory

This variable points to the directory on the local machine where the bitmap images reside. The format of the bitmap path must be as follows:

`\directory\subdirectory1\subdirectory2`

Default is:

C:\Program Files (x86)\ABB Industrial IT\Inform IT\Display Services\Client\Bitmap

Log Files Directory

If logging to file, this is the directory where the log files are placed. The default is:

C:\Users\<current windows User>\AppData\Roaming\ABB\IM\AID\Log

Communication, Timeout in Seconds

The timeout value (in seconds) determines how long the client waits for reply from the server. If you use a slow speed connection such as a modem to access the server, it may be necessary to increase the timeout value, for example to 60 seconds.

Application Start-up

Start-up of the Display Server is under control of the Process Administration Services. This is described in *System 800xA Information Management Configuration (3BUF001092*)*. Display clients may be started from the [Windows](#)

[task bar](#), or you can [create a shortcut](#) on your desk top. If you are using the client to build a display, be sure to start the client in the Build mode when you log in.

Starting the Display Client from the Task Bar

To start a PC client, from the task bar, choose: **ABB Start Menu>ABB Industrial IT 800xA>Information Mgmt>Display Services>Client> IM Display Client**
This opens the Display Services log in dialog, [Figure 3](#).

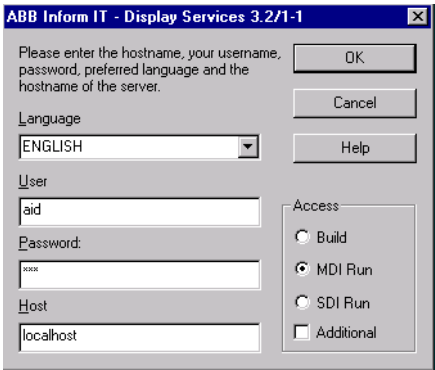


Figure 3. Display Services Login Dialog

Enter the required information as described in [Table 1](#), then click **OK**.

Table 1. Login Information

| Field | Description |
|----------|---|
| Language | This field indicates the language file for the user interface. You can use the pull-down menu to select a different language. To create additional language files, refer to the section on user preferences in <i>System 800xA Information Management Configuration (3BUF001092*)</i> . |
| User | Enter your user name in this field. Users are not defined by default and must be created. Refer to the <i>Configuring User Preferences</i> section in <i>System 800xA Information Management Configuration (3BUF001092*)</i> . |

Table 1. Login Information

| Field | Description |
|----------|---|
| Password | Enter your user password in this field. You can assign new passwords as required. Refer to the <i>Configuring User Preferences</i> section in <i>System 800xA Information Management Configuration (3BUF001092*)</i> . |
| Host | Enter the <i>computer name</i> for the computer where the Display Server software is installed. To find the name, go to that computer. From the Windows task bar, open: Control Panel , and double-click Network. This displays the Network dialog. Click the Identification tab to see the Computer Name. |
| Access | The access mode determines what functionality you will have access to: Build - This gives you access to both build and runtime functions. MDI Run- Multiple Document (Display) Interface. This gives you runtime access, and lets you run multiple displays at the same time. SDI Run- Single Document (Display) Interface. This also gives you runtime access; however, you can only run one display at a time. Additional - This is for starting applications that use Display Services as a container (meaning that the application's objects and controls are accessible from the Display Services Object Browser). |

Creating a Shortcut for Starting the Display Client

This lets you avoid having to log in explicitly each time you start the client. To create a shortcut:

1. Use the Windows Explorer to navigate to the client executable file (Aid.exe). The path is:
c:\Program Files\ABB Industrial IT\Inform IT\Display\ Services\Client\Bin
2. Right click on Aid.exe and choose **Create Shortcut** from the context menu, [Figure 4](#).
3. Drag the shortcut onto your desktop. Then right click on the shortcut and choose **Properties** from the context menu. This displays the Properties dialog.
4. Click the **Shortcut** tab, and use the Target field to specify your login parameters, [Figure 5](#). Enter the parameters after the target text string (enclosed in quotation marks). The parameters you can specify are described in [Table 2](#).

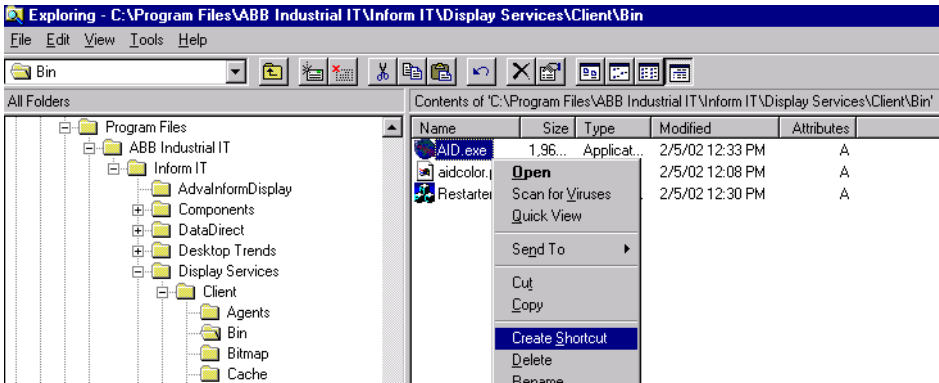


Figure 4. Creating a Shortcut for Aid.exe

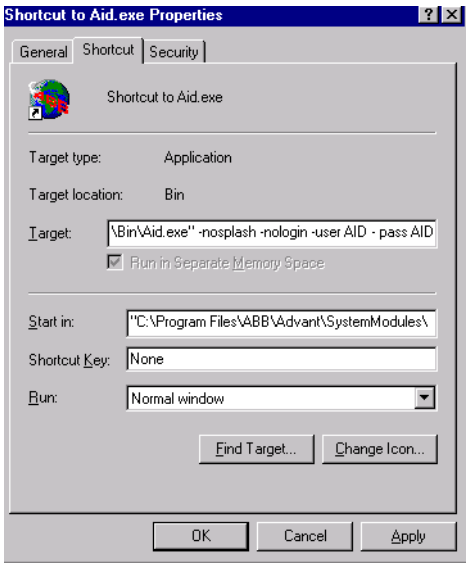


Figure 5. Specifying Login Parameters for the Shortcut

Table 2. Parameters for Display Client Shortcut

| Parameter | Description |
|-------------------------|--|
| -nosplash | This parameter is optional. If specified, the splash screen is not shown. This can be used with both login dialog and Command line login. |
| -nologin | If this parameter is specified, the login dialog is not shown. The remaining parameters described below are only applicable when the -nologin option is specified. If a parameter is omitted, its default value is used. |
| -user <username> | This is the user name used to log into Display Services when you launch the Display client. The default is the Windows user name. Unless your display user name is the same as your Windows user name, this parameter must be specified. |
| -pass <password> | This is the password used to log into Display Services when you launch the Display client. The default is blank (no password). This parameter must be specified. |
| -host <TCP/IP or alias> | This is the TCP/IP address or alias of the server to which the client will be connected. The default is: localhost |
| -port <portnumber> | This parameter is optional. This is the socket port number of the server to use for connection. The default is: 19014 |
| -lang <language> | This parameter is optional. This is the language to use for translation. The default is: English |
| -mode <access mode> | This parameter is optional. It specifies the access mode. The choices are: Build, MDI, and SDI. The default is: SDI |

To launch the display client from the shortcut without having to log in, define the shortcut parameters as follows:

-nologin -user *your username* -pass *your password*
-host *your hostname* -mode build

Checking Display Server Status

To check server status from a PC platform, from the task bar choose: **ABB Start Menu> ABB Industrial IT 800xA>Information Mgmt>Display Services >IM**

Display Server Status. This displays a dialog for specifying the server hostname, [Figure 6](#).

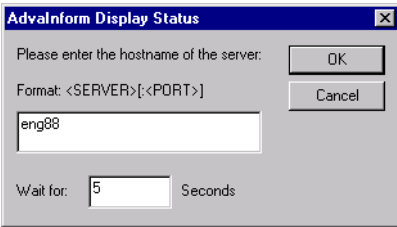


Figure 6. Specifying Server Hostname

Enter the hostname. As an option you can specify the maximum time to wait for the server to respond. Click **OK**. This displays the server status window, [Figure 7](#). Refer to [Table 3](#) for details.

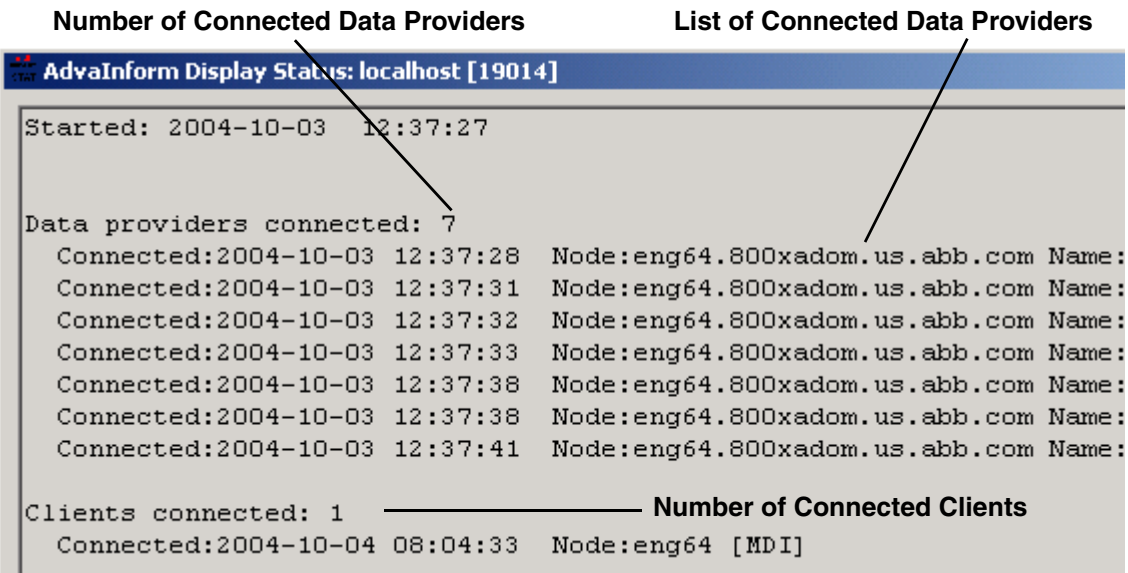


Figure 7. Display Services Server Status Window

Table 3. Server Status Description

| Server Status | Description |
|--------------------------|--|
| License Text | After you enter your permanent license as described in Setting Up PC Client Environments on page 21, this field should show your permanent license text. |
| License Will Expire | This field indicates when the current license is scheduled to expire. After you enter your permanent license as described in Setting Up PC Client Environments on page 21, this field should indicate your permanent license will never expire. |
| Started | This field indicates when the server was started. |
| Licenses | <p>These fields indicate the total number of licenses (how many you purchased), and the available licenses (not currently being used) for the following categories:</p> <ul style="list-style-type: none"> • Build - When you log in with Build access you can use Display Services both in the Build mode and the Run mode. • MDI - Multiple Document (Display) Interface. When you log with MDI Run access, you can run multiple displays at the same time. • SDI - Single Document (Display) Interface. When you log with SDI Run access, you can only run one display at a time. • ADD - Information Manager DataDirect. • DP - This is the number of data providers. For further information regarding data providers, refer to the section on data providers in <i>System 800xA Information Management Configuration (3BUF001092*)</i>. |
| Facilities | This is not applicable at this time. |
| Data Providers Connected | This shows the number of data providers connected. Display Services must have at least one data provider connected. |
| Clients Connected | This shows how many clients are connected to this server. The information provided includes the node name, type (MDI or SDI as described in Licenses above), and date and time when the client connected. |

Stopping the Display Server

Shutdown of the Display Server is under control of the Process Administration Services. This is described in *System 800xA Information Management Configuration (3BUF001092*)*.

Launching the DisplayIE Client as an ActiveX Control

This section provides guidelines for configuring an external application such as Visual Basic or Internet Explorer to launch the DisplayIE Client as an ActiveX control. You must install and license the DisplayIE option to use this functionality.

An example web page, [Figure 8](#), is provided in:

c:\Program Files\ABB Industrial IT\InformIT\DisplayIE\Examples\Web

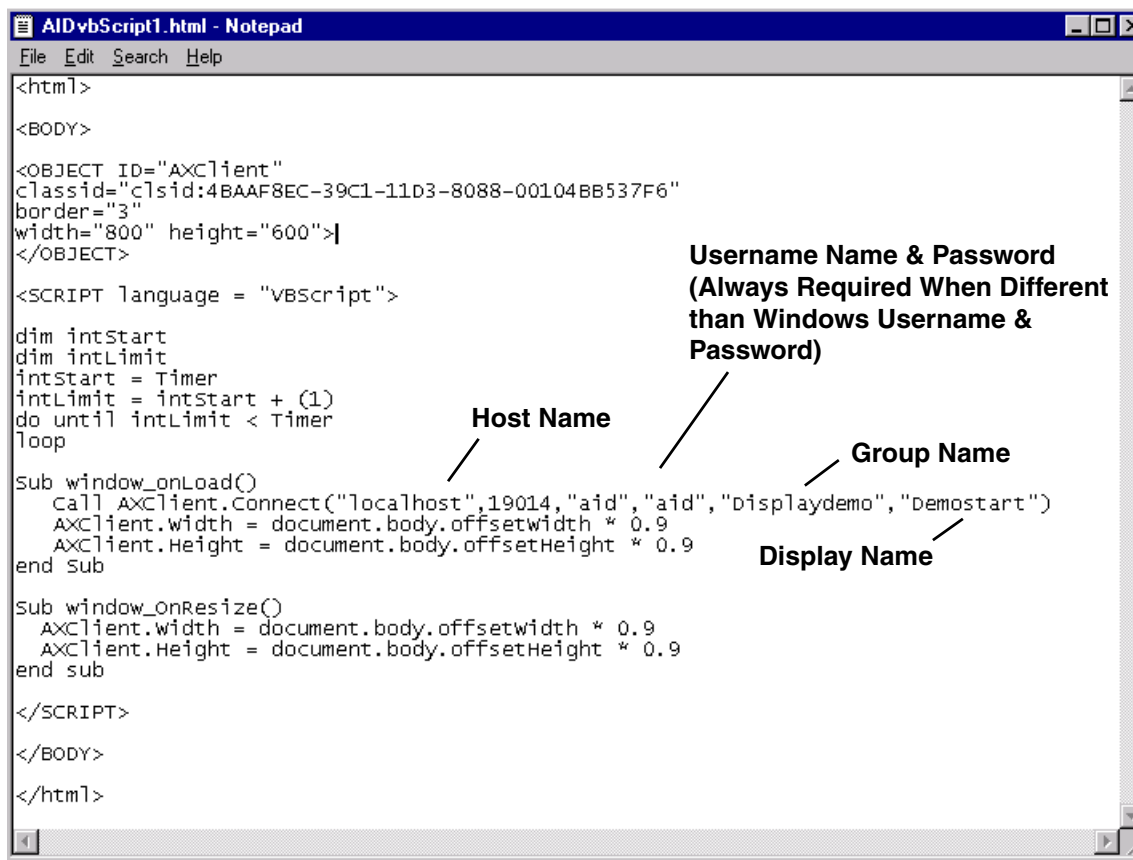


Figure 8. Example HTML Page

You can use this page to launch a display by editing the host name, group name, and display name parameters shown in [Figure 8](#).

The following configuration parameters are supported:

- Methods
 - [Setup](#)
 - [Connect](#)
 - [Disconnect](#)
- Event
 - [LostConnection](#)
- Property
 - [Version](#)

Setup

Syntax *Setup*(String <DisplayPath>, String <BitmapPath>, String <LanguagePath>, short <CommTimeOut>, short <DispCache>, long <UECacheSize>, String <Language>);

If you use Setup, it must be called before [Connect](#). If you do not use Setup, the default values will be used. See [Table 4](#).

Table 4. Arguments for Setup

| Argument | Default | Description |
|--------------|------------|---|
| DisplayPath | ..\Cache | The directory to place cached displays. Used if not "". |
| BitmapPath | ..\Bitmap | The directory where bitmap files are placed. Used if not "". |
| LanguagePath | ..\Lang | Path to directory containing language file. Used if not "". |
| CommTimeOut | 30 seconds | Communication timeout towards the server. Used if > 0. |
| DispCache | 0 | If 0 display caching is disabled, otherwise display caching is used. |
| UECacheSize | 0 | Specifies the amount of memory to use for caching user elements (in bytes), if 0 user element caching is disabled. |
| Language | ENGLISH | Specifies the language file to use. For ENGLISH, just specify "". All the default paths are relative to where the client is installed. All user specified paths must be absolute. |

Example:

Setup client to use display caching at an alternative directory and timeout 60 seconds:

```
call AXClient.setup("d:\AXCache","","",60,1,0,"")
```

Connect

Syntax *Connect*(String <Host>, long <Port>, String <User>, String <Password>, String <Group>, String <Display>);

Table 5. Arguments for Connect

| Argument | Default | Description |
|----------|-----------|---|
| Host | localhost | TCP/IP address or alias of server, "" = localhost. |
| Port | 19014 | Port address to use, typically 19014. Used if > 0, otherwise 19014. |
| User | None | Username to use for login. If "", the Windows user login name is used. |
| Password | None | Password to use for login. If User is Windows user (or ""), the password is not required. |
| Group | None | The name of the Group in which the startup display exists. Must be specified. |
| Display | None | The name of the Display which should be opened at startup. Must be specified. |

Example:

Connect to the local machine, using the Windows user as login name:

```
call AXClient.connect("localhost",19014,"","","Displaydemo",  
"Demostart")
```

Disconnect

Syntax *Disconnect*();

Takes no arguments. When leaving a web page the client is disconnected automatically, but can be used to manually disconnect.

LostConnection

The LostConnection event is fired when the control loses connection to the server.

Version

Syntax String <Version>;

The Version property returns the version of the AXClient control as a string formatted like: "3.1/0 Beta 2". It is read only, trying to assign it will have no effect.

Example:

Retrieve the version from VB code:

```
Dim VersionString as String
VersionString = AXClient.Version
```

Section 3 A Quick Tutorial

Introduction

This section provides two quick lessons for learning how to use Display Services.

The first lesson, [Building a Basic Display for Process Data](#) on page 35, takes you step-by-step through the procedure for building a dynamic display for viewing process data. This lesson shows you how to:

- create a new display.
- draw and position display elements on the display.
- use both interactive dialogs and scripts to configure the display element properties to support the required data retrieval and display functionality.

The second lesson, [Building a Trend Display](#) on page 46, demonstrates how to retrieve and display historical data. This lesson does not cover the basic procedures such as inserting a display element in the same detail as lesson 1. Therefore, it is recommended that you complete the first lesson before attempting the second. Each of these lessons can be completed in about ten minutes or less.

Building a Basic Display for Process Data

The display you are going to build in this lesson displays a process value whose source is an OPC item, for example a tag in an AC 800M controller or a softpoint. This simple application requires a Numeric display element to query the process object and display the value.

Start the Display Client in Build mode as described in [Startup](#) on page 21. Then proceed with [How to Create a New Display](#) on page 36.

How to Create a New Display

Each display you create must belong to a display group. Grouping is a means of organizing related displays. Each display must have a unique name within the group.

To create a new group:

1. Select the display server icon in the Object Browser. Then right click on the display server icon, and choose **Add child > GROUP** from the context menu, [Figure 9](#).

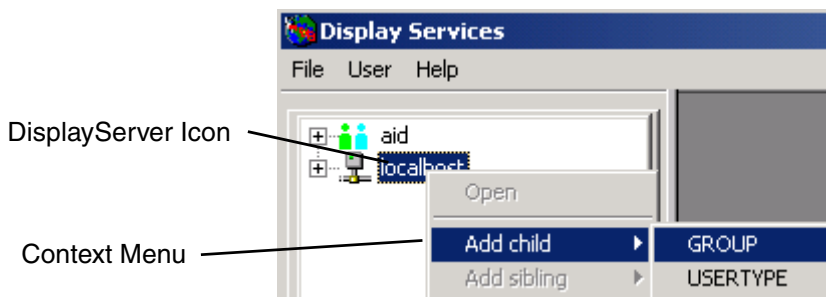


Figure 9. Creating a New Group

This displays the New Leaf dialog for specifying the Group Name, [Figure 10](#).

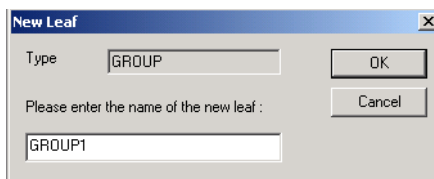


Figure 10. Dialog for Specifying Group name

2. Enter a unique group name, then click **OK**.
3. Add a new display to the group. Select GROUP1 in the Object Browser. Then right click on GROUP1, and choose **Add child > DISPLAY** from the context menu, [Figure 11](#).

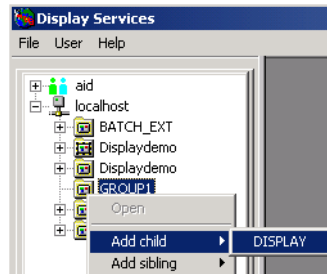


Figure 11. Adding a Display to a Group

This displays the new Leaf dialog again, this time for entering the name of the new display. Enter a name for the new display, [Figure 12](#).

Names in Display Services are case sensitive. Also, you can not use blank characters in the display name. The display name must be unique within the group. If the name is not unique in the group, an error message is displayed.

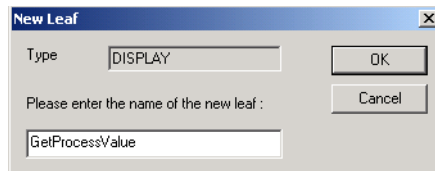


Figure 12. Entering a Display Name

4. Click **OK** when you are finished.

This opens a new display in the display area, and activates the Display Element toolbar, [Figure 13](#).

5. Next, add the [Numeric display element](#) to the new display.

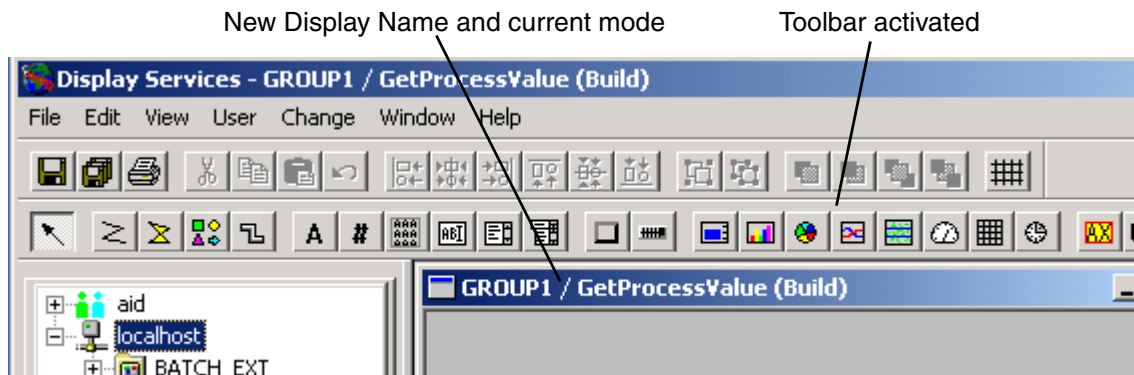


Figure 13. New Display

How to Insert a Display Element

The numeric display element will query the aspect object property and retrieve the process value. The query is channeled through the system's OPC DA server. All queries for OPC-based data are made using the display scripting language *Data* statement which can only be entered via certain display elements. To insert the Numeric display element:

1. Click on the Numeric button, [Figure 14](#).

Icon for Numeric Display Element

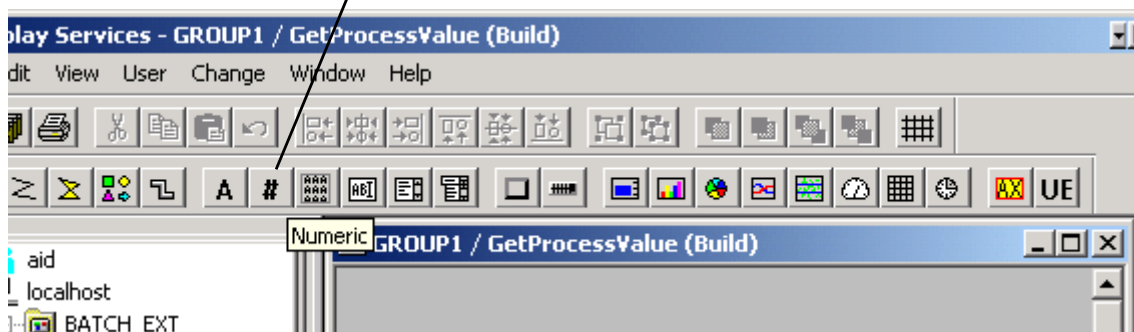


Figure 14. Numeric Button

2. Move the cursor into the display area, [Figure 15](#). When the cursor is over the display, it becomes a crosshair.

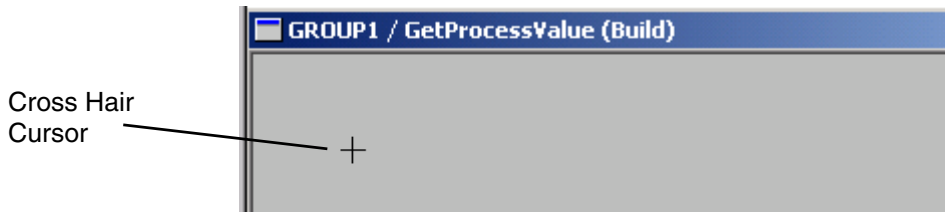


Figure 15. Crosshair Cursor

3. Move the crosshair to the approximate location where you want to place the Numeric display element. For this tutorial, place the Numeric display element near the upper left corner of the display.

When the crosshair is positioned where you want it, press and hold the left mouse button, and drag the mouse in a diagonal direction down and to the right. You will see a box that expands in width and height as you drag the mouse.

This gives you an idea of the size of the display element, [Figure 16](#).

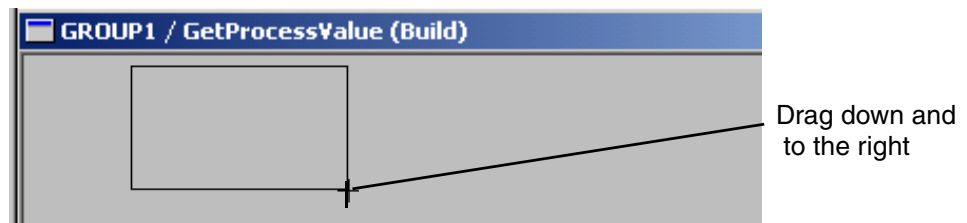


Figure 16. Sizing the Numeric Display Element

4. When the box is approximately the size that you want for the Numeric display element, release the left mouse button. Now you will see the Numeric display element, [Figure 17](#).

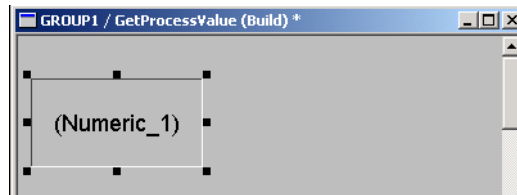


Figure 17. Numeric Display Element

You can re-position and re-size the display element later, either by clicking and dragging, or by re-defining the display element properties. For instance, if you want to move the display element, simply click on it and drag. If you want to shrink or expand the display element, click on one of the handles (small black boxes) that surround the display element and pull to expand, or push to shrink.

Next, [configure the properties for the numeric display element](#).

How to Configure Display Element Properties

You configure the properties of a display element via the Properties dialog. To configure the properties for the Numeric_1 display element:

1. Display the Properties dialog for the Numeric display element.
 - a. Select the display element.
 - b. Right click on the selected display element to display the context menu, then choose **Properties**, [Figure 18](#).

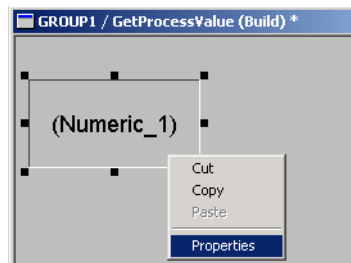


Figure 18. Displaying the Properties Dialog

The Properties dialog is shown in [Figure 19](#). There are three sets of properties listed under three different tabs: **Members**, **Methods**, and **Events**. The properties under the selected tab are shown in the left pane - property list. The selected property may be edited in the right pane - property definition window.

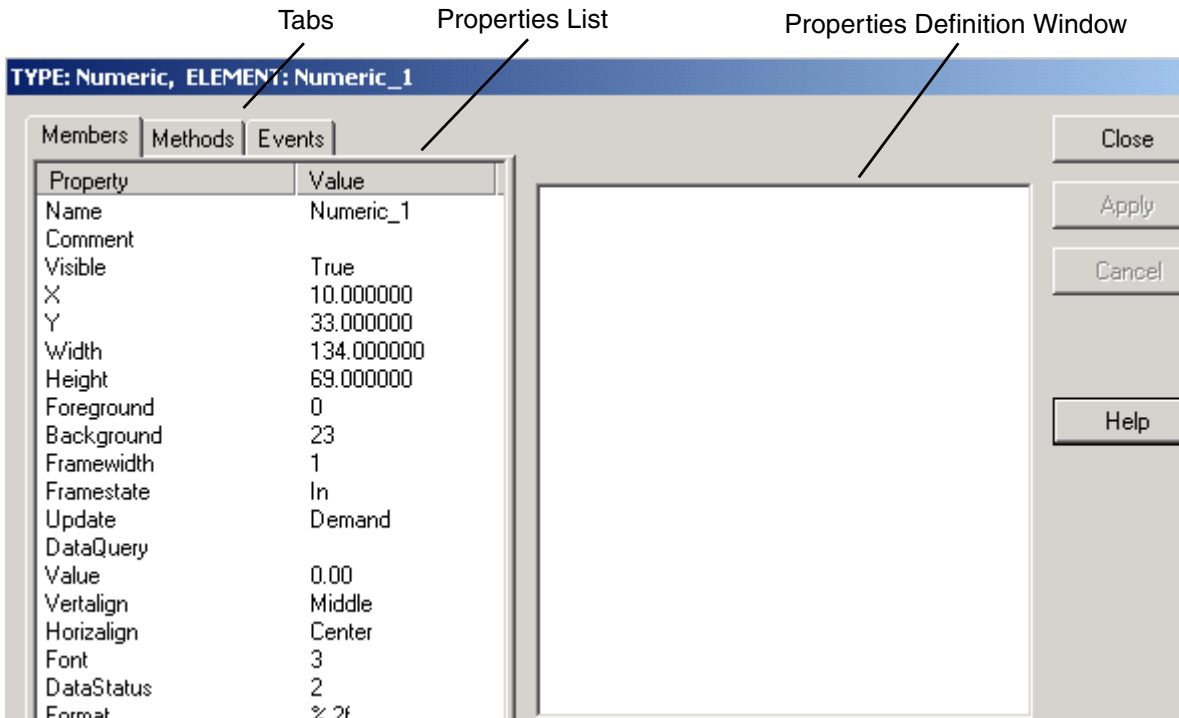


Figure 19. Properties Dialog Box for Numeric Display Element

The size and position are defined according to how you placed the display element in the display. For this tutorial, the properties that you need to define are listed under the **Members** and **Events** tabs. The **Members** tab is selected by default.

[Table 6](#) indicates which properties need to be changed to create the Numeric display element for this tutorial. Any property not listed in the table can be left at its default value. To learn the purpose of any properties not covered in this tutorial, refer to [Working with Display Elements](#) on page 102.

Now edit the properties listed in [Table 6](#).

Table 6. Numeric_1 Properties

| Property | Value | Comment |
|-----------|--|---|
| Name | Numeric_1 | This is the default name. |
| X | 120 | This is the X coordinate of the upper left corner of the display element. |
| Y | 120 | This is the Y coordinate of the upper left corner of the display element. |
| Width | 100 | |
| Height | 25 | |
| DataQuery | See How to Define a Query on page 43 | Gets value from aspect object property. |

2. Click on the X property. When this property is selected, the current value is displayed in the Properties Definition window, [Figure 20](#).

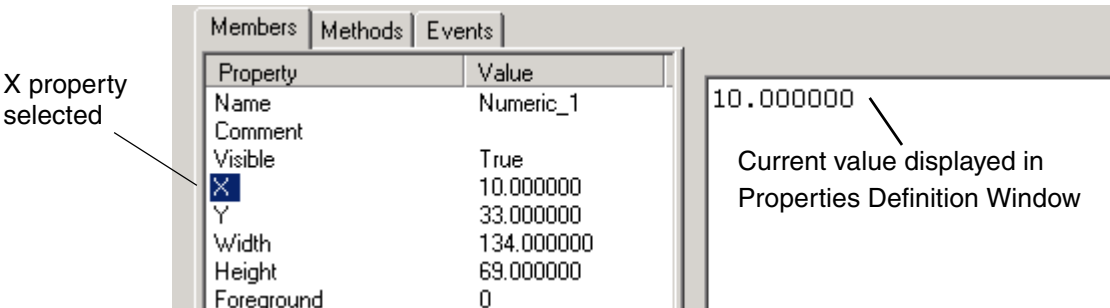


Figure 20. Editing a Property

3. Click inside the window, change the value to 120, and then click **Apply**. After you change the value, notice that the horizontal position of the Numeric display element shifts according to the new X coordinate.
4. Repeat this procedure for the other properties.
- Next [define a query for the Numeric display element](#).

How to Define a Query

This query will return the value of a specified aspect object property. The query must also specify which data provider will be used to retrieve the data, and the rate at which the object will be queried. To enter this query (reference [Figure 21](#)):

1. Click on the DataQuery property in the property list, and enter the following query in the property definition window. You may have to expand the Properties dialog to show the entire query. Enter:

#data ("AIPOPC", "subscribe", "tagname", dcs_3s);

where *tagname* is the name of the aspect object property. If you do not know the name, you may use the OPC object browser to find the object in the Aspect Directory, and then copy and paste it into the query. Instructions for using this browser are in *800xA Information Management Data Access and Reports*.

2. DataQuery entered in the property definition window
Keywords such as data and dcs_3s are indicated in blue

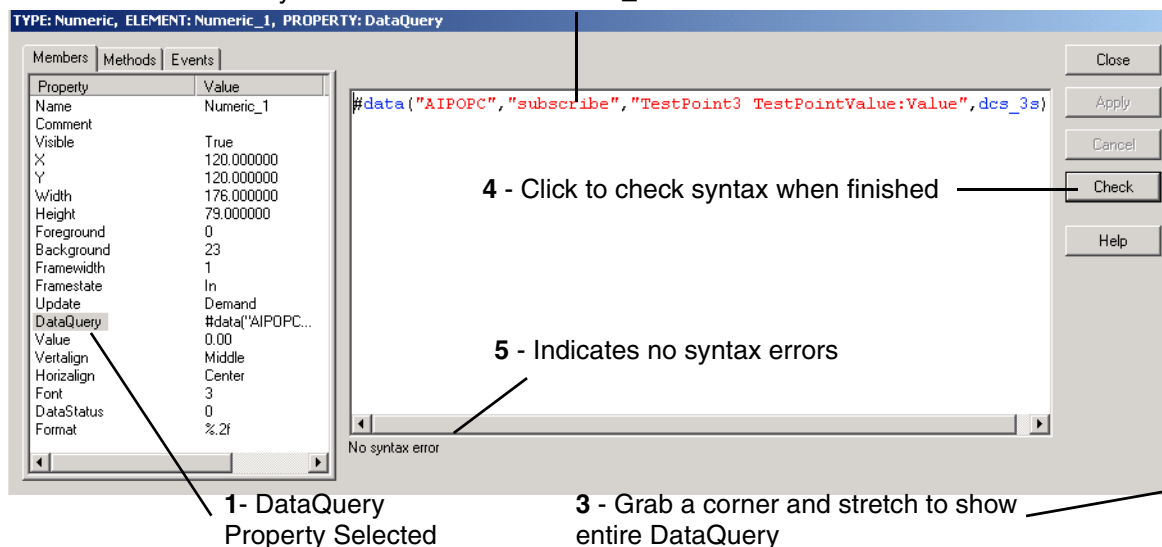


Figure 21. Numeric_1 DataQuery

This query uses the *data* statement which is required for OPC-based data. A complete description of the data statement is provided in [data](#) on page 295. To specify the data statement specifically for this tutorial:

- **data** - This is the keyword for the data statement.
- **"AIPOPC"** - This specifies the data provider which will be used to retrieve the data. Enter the name in double quotes. This data provider must be operational (default state). If you suspect the data provider is not running, refer to the section on data providers in *System 800xA Information Management Configuration (3BUF001092*)*.
- **"subscribe"** - This indicates data will be read from the specified aspect object property (rather than written). Enter this word in double quotes. **Note:** As an option, you can query for the object's last history value with **SubscribeLHV** if the object has an associated history log.
- **"tagname"** - Enter the name of the aspect object property in double quotes, for example: **"TestPoint3 TestPointValue:Value"**
- **dcs_3s** - This indicates the rate at which the aspect object property will be queried - in this case, every 3 seconds.



Keywords such as **data** and **dcs_3s** are indicated in blue. The remainder of the text will be shown in red if entered correctly.

2. Click **Apply** to save the query, and then click **Check** to check the syntax. The result is indicated on the message line near the bottom of the dialog. In [Figure 21](#), the syntax checker indicates no errors.

This completes the property definition for Numeric_1. Next [save your work](#).

Saving Your Work

Be sure to save your work. Click the save icon on the toolbar, [Figure 22](#), or choose **File > Save** from the main menu bar.



Figure 22. Save Icon

This concludes the build portion of this tutorial. At this point you can either close the build window, or leave it open as you test the display in Run mode. Typically,

you should leave the build window open so that you can make changes in the event that you find an error during the test. Now, [test the display in Run mode](#).

How to Test Your Display

To launch the display in Run mode, you can use either of the following methods:

- Hold down the CTRL key and double-click the display icon in the Object Browser.
- Select the display icon in the Object Browser, right click and choose **Run** from the context menu, [Figure 23](#). Either way the display is opened in the Run mode.

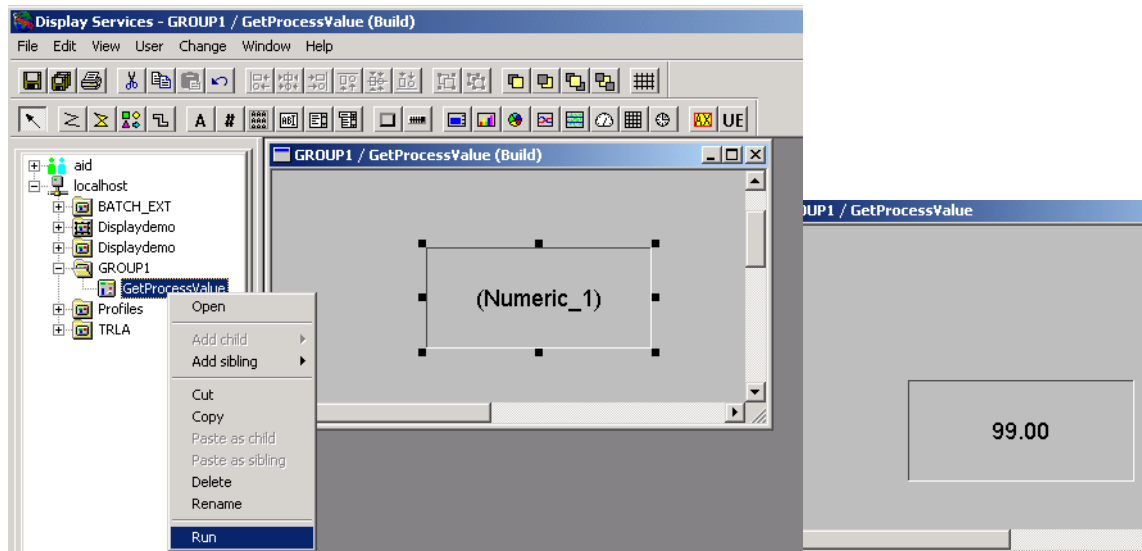


Figure 23. Running the Display

The Numeric display element may briefly show a red cross while data is being redivide. This cross will be replaced with a value if the display elements and data provider are properly configured and operating correctly. If the aspect object property that you specified is actively changing, you'll see the value update at the specified rate (every three seconds).

Finish the tutorial by [exiting Display Services](#).

How to Leave the Run Mode and Exit Display Services

Close the build and runtime windows. To do this, from the main menu bar choose **Windows > Close All**, [Figure 24](#).

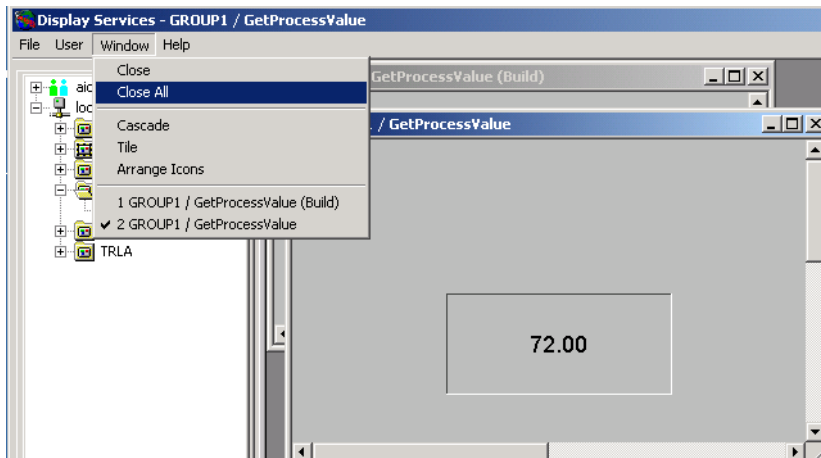


Figure 24. Closing the Build and Runtime Windows

If there are any changes that you have not saved yet, you will be prompted to save now.

To exit Display Services, choose **File > Exit**.

Building a Trend Display

The display you are going to build in this lesson displays trend data for a property log that you specify at runtime. This application requires a Trend display element to display the trend data, an Edit display element that lets you enter the property log name, and a Button display element to update the Trend display element with the new request and cause it to issue a new query.

To build this display:

1. Start the Display Client in Build mode as described in [Startup](#) on page 21.
2. Create a new display under GROUP1. (GROUP1 was created for lesson 1 as described in [How to Create a New Display](#) on page 36.). To do this:

- a. Select GROUP1 in the Object Browser. Then right click on GROUP1, and choose **Add child > DISPLAY** from the context menu, [Figure 25](#).

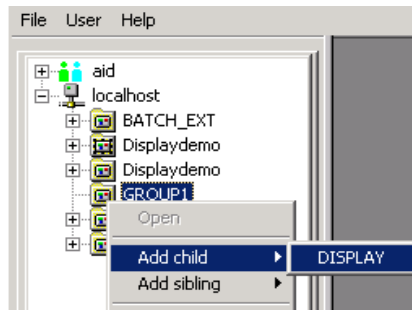


Figure 25. Adding a New Display to the Group

- b. Enter a name for the new display, [Figure 26](#), then click **OK**.

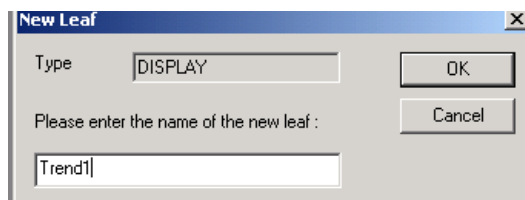


Figure 26. Entering a Display Name

3. Insert a Trend display element. To do this:
 - a. Click on the Trend button, [Figure 27](#).

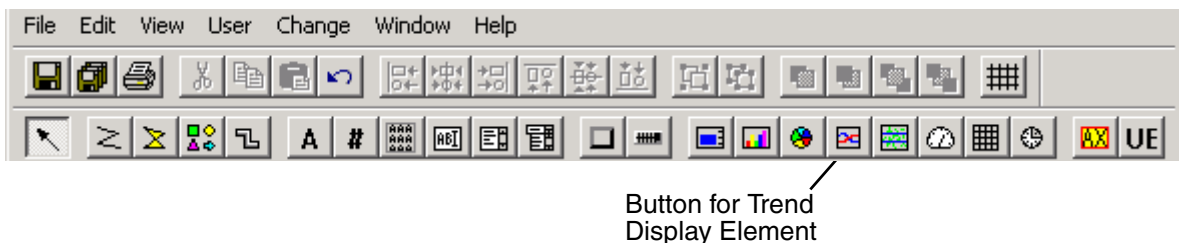


Figure 27. Trend Button

- b. Place this display element near the upper left corner of the display. The Trend display element is shown in [Figure 28](#).

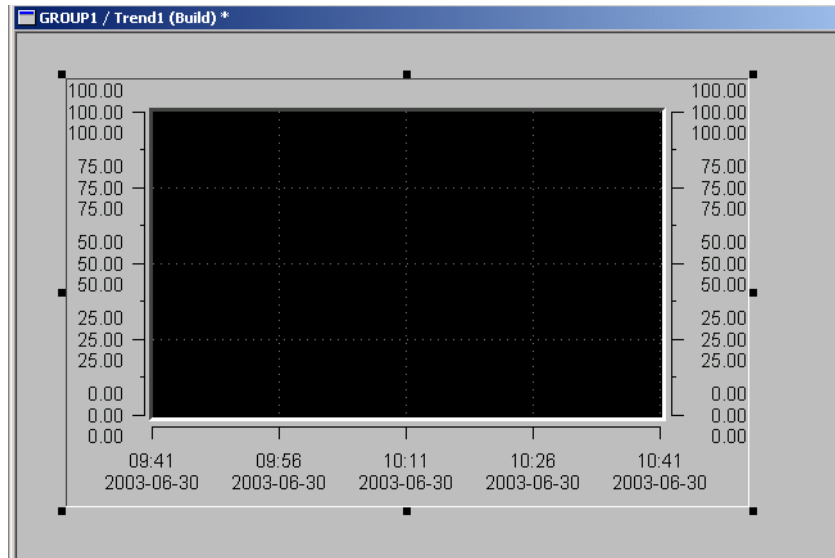


Figure 28. Trend Display Element

4. Configure the properties for the Trend_1 display element. Some properties are configured for the display element itself, while others are configured on an individual basis for each trend. These properties have a **Tn_** prefix, for example, **T1_LogName**. All properties that you need to configure for this lesson are on the **Members** tab.

To request data from a property log via the 800xA system OPC HDA server, you must:

- specify the data source as being an OPC server (OPCHDA)
- specify the name of the data provider which supports data retrieval via the PPA OPC HDA server (AIPHDA)

The default color for all traces is black which is the same as the trend background color. Thus traces will not be visible unless you change the color for each trace.

You may also be required to adjust the scaling parameters, depending on the value range of your log. The default range is 0-100. You are not required to specify the log name since this will be supplied via the Edit field at runtime.

To configure the Trend display element properties:

- a. Display the Properties dialog for the Trend display element, [Figure 29](#).

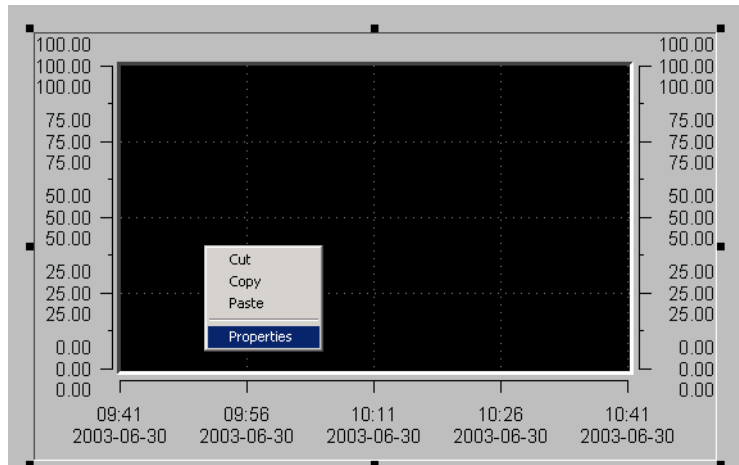


Figure 29. Displaying the Properties Dialog

- b. Configure the properties as indicated in [Table 7](#). You may use the default values for all other properties.

Table 7. Trend_1 Properties

| Property | Value | Comment |
|-------------|----------|---|
| Update | Cyclic 3 | Causes the trend display to update every three seconds |
| Data Source | OPCHDA | For data retrieval from an OPC server, in this case, the 800xA system OPC HDA server. |
| T1_LogName | AIPHDA | This is the name of the data provider which supports data retrieval from the 800xA system OPC HDA server. |

Table 7. Trend_1 Properties

| Property | Value | Comment |
|-----------------------------|----------------|--|
| T1_Scalemin, T1_Scalemax | | Adjust as required based on the value range of your log |
| T1_Color | 65535 (Yellow) | You can enter the color directly, or use one of the two palettes provided (Windows palette is recommended). Colors are numerically coded (65535 = yellow). If you do not know the correct number, click the Palette button. This displays the Windows Color Palette. Choose a color from the palette. |

- c. Click **Apply** to apply the changes, then click **Close** to close the Properties dialog.
- 5. Insert an Edit display element below the Trend display element. The Edit button on the tool bar is shown in [Figure 30](#).



Figure 30. Edit Button

- 6. Configure the properties for the Edit_1 display element. The purpose of this field is simply to let the display user enter the name of the property log whose trend data they wish to display. The properties to be configured for this display element are for positioning and size, and are located on the **Members** tab. Guidelines are provided in [Table 8](#). You may need to adjust these values depending on the size and position of the Trend display element.

Table 8. Edit_1 Properties

| Property | Value | Comment |
|----------|-------|---|
| X | 292 | Places the button slightly indented relative to the Trend display element |
| Y | 430 | Places the button slightly below the Trend display element |
| Width | 400 | Make sure the Edit field is long enough to hold the log name |
| Height | 35 | |

Click **Apply** to apply the changes, then click **Close** to close the Properties dialog.

7. Insert a Button display element to the left of the Edit display element. The Button button on the tool bar is shown in [Figure 31](#).

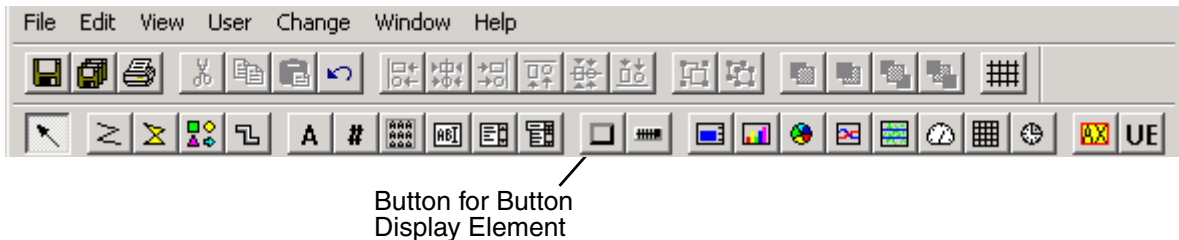


Figure 31. Button Button

8. Configure the properties for the Button_1 display element. The main property to be configured is the Action property on the **Events** tab. This action requires a script to clear any trend data that may exist from a past query, write the property name in the Edit field to the T1_ObjectName property of the Trend Display element, and then cause the Trend display element to query the specified log. These actions will be performed when the button is pressed. There are also some properties related to positioning, size, and appearance on the **Members** tab. These may have to be adjusted, depending on the size and position of the Trend and Edit display elements. Details are provided in [Table 9](#).

Table 9. Button_1 Properties

| Property | Value | Comment |
|------------|-------|---|
| X | 112 | Places the button slightly to the left of the Edit display element |
| Y | 430 | Aligns the button vertically with the Edit display element |
| Width | 150 | |
| Height | 35 | Matches the height of the Edit display element |
| FrameWidth | 5 | Gives button depth |
| Caption | blank | Clear the default caption (Button) so that no text will appear on the button in Run mode |
| Action | | This property is located on the Events tab. See Button Action on page 52. |

Button Action

Click the **Events** tab, and then click the Action property. Enter the following script for this property ([Figure 32](#)):

```
$Trend_1.TrendMode = "ClearLogData";  
update (Trend_1);  
  
$Trend_1.T1_ObjectName = "I" + $Edit_1.Value + "I";  
  
$Trend_1.TrendMode = "QueryValues";  
update (Trend_1);
```

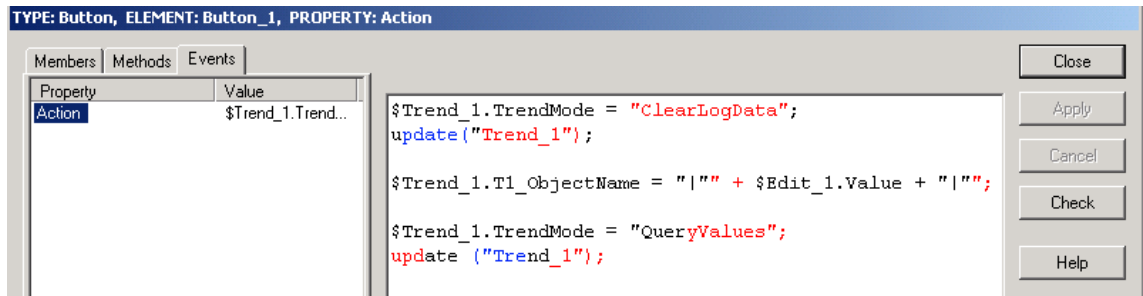


Figure 32. Button_1 DataQuery

Lines 1 & 2 clear any trend data that may currently be on the display.

Line 3 writes the contents of the Edit field to the T1_ObjectName property of the Trend_1 display element. This syntax must be followed closely to ensure the name is written embedded in quotation marks. This syntax is described further in [Strings](#) on page 281.

Lines 4 & 5 cause the Trend display element to query the specified object for data. For further information regarding the TrendMode property, see [TrendMode](#) on page 186.

When you are finished with the query, check the syntax, then click **Apply** to save the query.

This concludes the build portion of this tutorial. The configured display should appear as shown in [Figure 33](#). At this point you can either close the build window, or leave it open as you test the display in Run mode. Typically, you should leave the build window open so that you can make changes in the event that you find an error during the test.

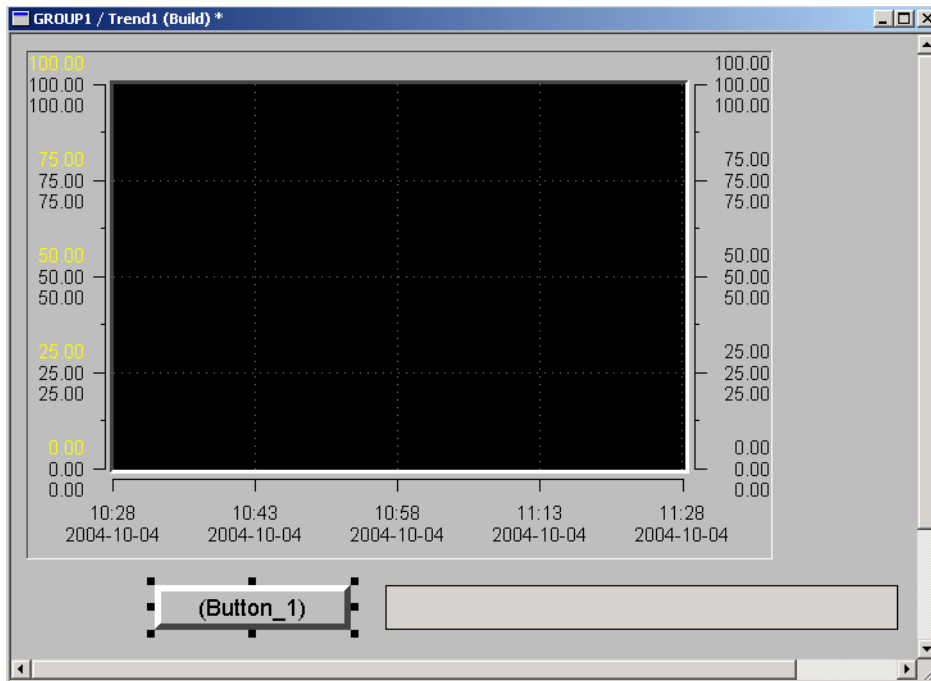


Figure 33. Configured Display

To launch the display in Run mode, select the display icon in the Object Browser, right click and choose **Run** from the context menu.

To display trend data (reference [Figure 34](#)):

1. Enter a log name in the Edit field. If you don't know the exact log name, you may use the OPC Browser as described in *800xA Information Management Data Access and Reports*.
2. Click the button to get the trend data.

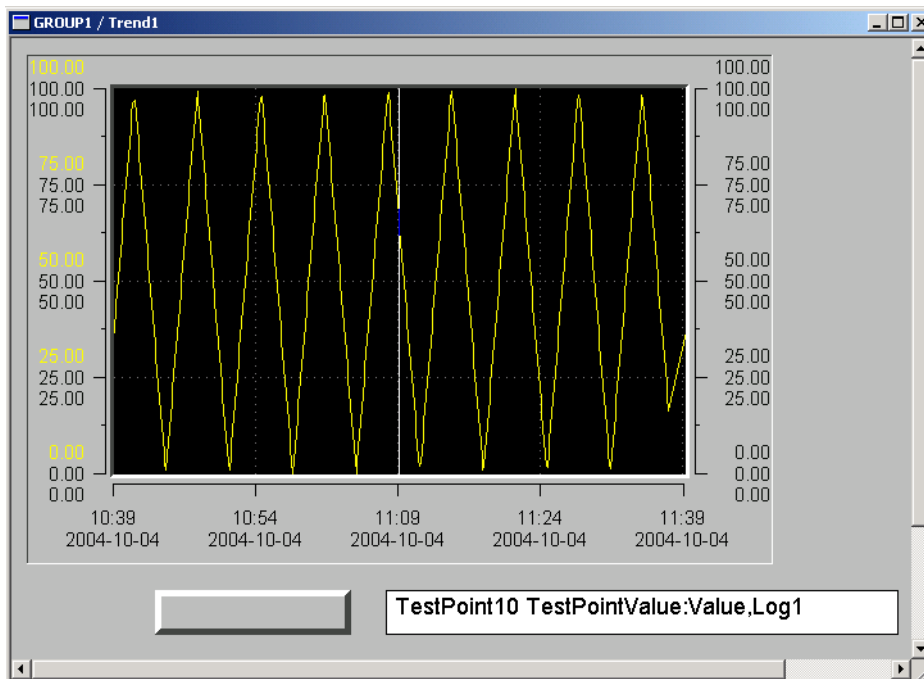


Figure 34. Trend Data Displayed

Close the build and runtime windows. To do this, from the main menu bar choose **Windows > Close All**.

To exit Display Services, choose **File > Exit**.

Section 4 User Interface & Application Tips

Introduction

This section briefly describes the Display Services [user interface](#), and provides some [application guidelines](#).

User Interface

All functions are executed via the main window, [Figure 35](#).

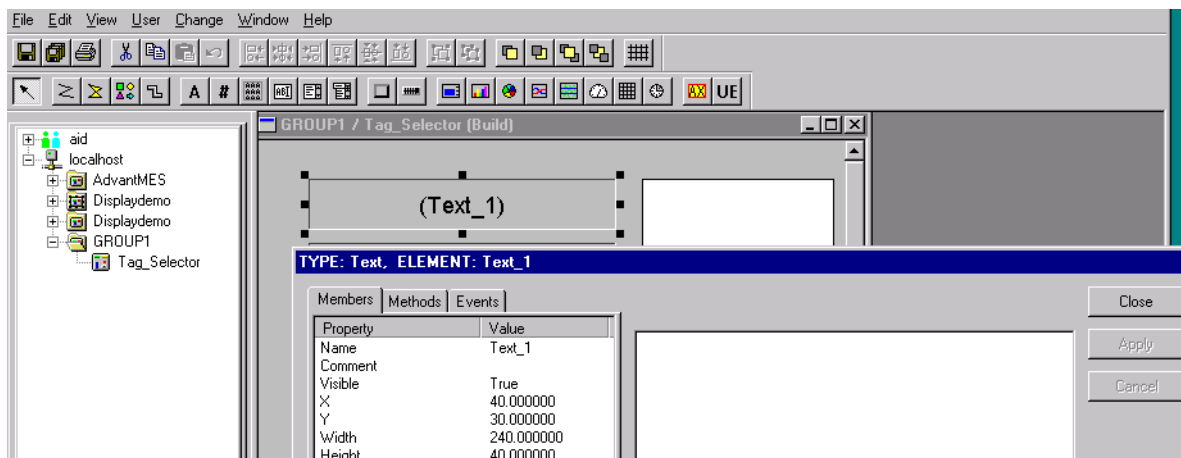


Figure 35. Main Window for Display Building

The user interface for this window is based on Windows technology. Knowing how to operate in the Windows environment is a prerequisite skill for using Display Services. This section describes user interface functions for Display Services:

- [Mouse Operation](#) on page 58
- [Object Browser](#) on page 58

- [Toolbar](#) on page 61
- [Main Menu Bar](#) on page 62

Mouse Operation

Virtually all operator interaction is via the mouse. Two mouse buttons are used. The left mouse button is used to operate the pull-down menus, buttons, tabs, list items, and edit fields. The right mouse button is used to display a context menu. For instance, you display the Properties dialog by first selecting a display element with the left mouse button, clicking on the display element with the right mouse button, and then choosing **Properties** from the context menu.

In procedures where you use the left mouse button, you are instructed to *click*. When you need to use the right mouse button, you are instructed to *right click*.

Object Browser

The objects that you build for your Display application (Display Groups, Displays, and User Elements) are represented as icons on the Object Browser tree, [Figure 36](#).

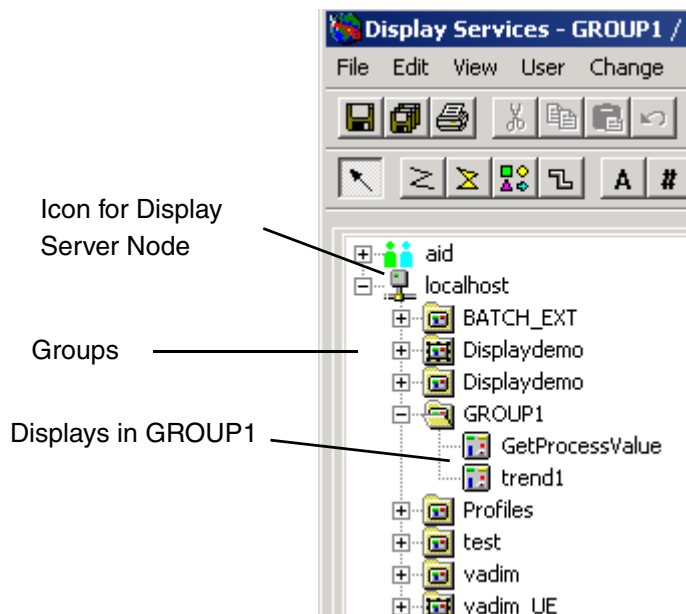


Figure 36. Display Objects in Object Browser

The Object Browser lets you add, remove, and otherwise manage Display objects, and invoke their respective controls. These functions are carried out via the Object Browser menu. To display this menu, select an object in the Object Browser and then right click. The available menu items vary depending on the icon you select. The menu items are: [Open](#), [Add Child](#), [Add Sibling](#), [Cut](#), [Copy](#), [Paste as Child](#), [Paste as Sibling](#), [Delete](#), [Rename](#), [Run](#).

Open

This opens the selected Display.

Add Child

This adds a child of the appropriate type (display, user element, group folder, or user type folder) under the selected icon:

- If you select the display server Icon, then **Add child** adds either a new group folder for grouping displays, or new user type folder for grouping user elements.
- If you select a group folder, then **Add child** adds a new display in the selected group.
- If you select a user type folder, then **Add child** adds a new user element in the selected user type.

When you add an object, the New Leaf dialog is displayed, [Figure 37](#).

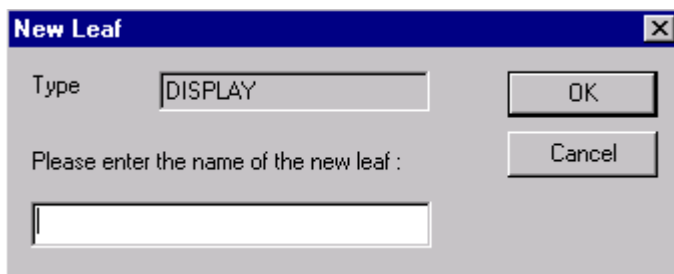


Figure 37. New Leaf Dialog for Adding a Recipe

The term *leaf* refers to a new icon (group folder, user type folder, display, or user element) in the Object Browser. This dialog has a read-only field that indicates the type of leaf to be added, and a field where you enter the name of the leaf to be added.

Type This is the type of leaf being added, for instance a display.

Name Enter the name for the leaf being added in this field.

Add Sibling

This adds a sibling of the same type for the selected icon. The dialog for Add sibling is the same as for [Add Child](#).

Cut

This removes the selected leaf from its current location in the Object Browser and places it in the clipboard so it can be pasted at another location in the Object Browser.

Copy

This places a copy of the selected leaf in the clipboard so it can be pasted at another location in the Object Browser.

Paste as Child

This pastes the contents of the clipboard as a child of the selected leaf in the Object Browser.

Paste as Sibling

This pastes the contents of the clipboard as a sibling of the selected leaf in the Object Browser.

Delete

This deletes the selected leaf from the Object Browser.

Rename

This displays a dialog for renaming the selected leaf.

Run

This opens the selected display in the run mode.

Toolbar

The toolbar contains buttons for inserting the various display elements, [Figure 38](#). To insert a display element, click the corresponding button, then use the left mouse button to specify the location and size. For an example, see [How to Insert a Display Element](#) on page 38.

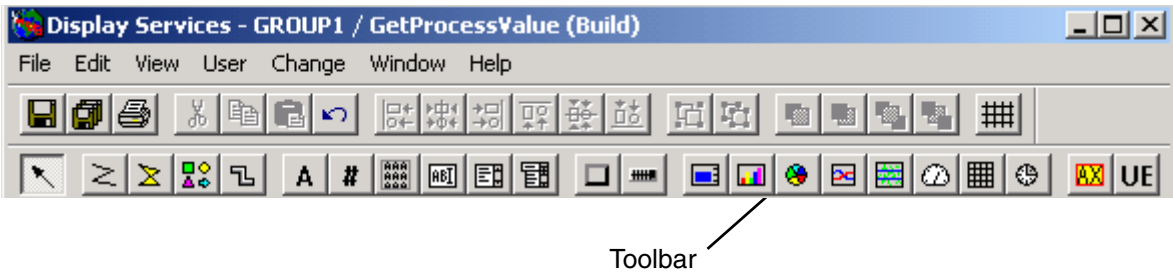


Figure 38. Tool bar

Main Menu Bar

The Main Menu, [Figure 39](#) is operated using the left mouse button.

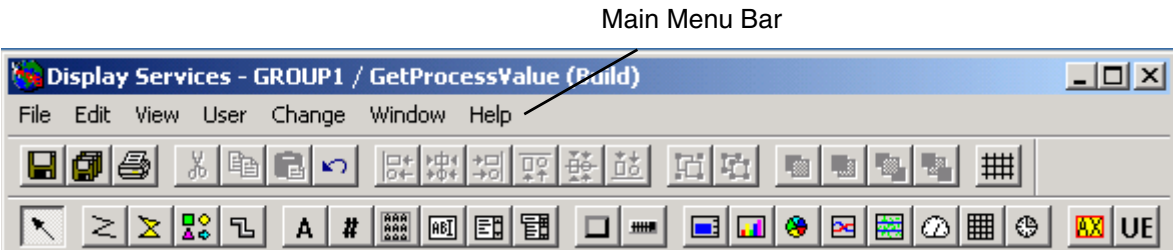


Figure 39. Main Menu




If a menu item is dimmed, it means that item can not be selected at this time. An arrow to the right of an item indicates that a cascade menu with more menu items will be displayed when this menu item is chosen. Three dots (...) means that a dialog box will be displayed when the menu item is chosen. The actions for some menu items can also be executed via keyboard short-cuts.

The menus are: [File](#), [Edit](#), [View](#), [User](#), [Change](#), and [Window](#).

File

The **File** menu contains menu items to save and print displays, and to exit Display Services, [Table 10](#).

Table 10. File Menu

| Name | Short Cut | Toolbar | Description |
|---------------|-----------|---|--|
| Save | Ctrl+s |  | Save the current display. |
| Save All | |  | Save all displays that are currently open. |
| Print | Ctrl+p |  | Print the current display. |
| Print Preview | | | Preview the current print selection. |
| Print Setup | | | Set up print options. |
| Exit | | | Exit Display Services. |

Edit

The **Edit** menu contains items for copying, cutting, deleting, and pasting display elements, and selecting all display elements in the display, [Table 11](#).

Table 11. Edit Menu





| Name | Short Cut | Toolbar | Description |
|------|-----------|---|---|
| Undo | Ctrl+z |  | Undo the last operation. The Undo stack is emptied when you save. |
| | Ctrl+x |  | Cut removes a selected display element from the display and puts it in the paste buffer. The removed display element can then be pasted in a different location in the same display, or it can be pasted in a different display. The cut display element remains in the paste buffer until you cut or copy another display element. This means you can paste the display element in more than one location |

Table 11. Edit Menu

| Name | Short Cut | Toolbar | Description |
|--------|-----------|---|---|
| Copy | Ctrl+c |  | Copy places a copy of the selected display element in the paste buffer. The copied display element can then be pasted in a different location in the same display, or it can be pasted in a different display. Unlike the Cut command, the selected display element remains unchanged (in its original location) on the display. The copied display element remains in the paste buffer until you cut or copy another display element. |
| Paste | Ctrl+v |  | Paste copies a display element from the paste buffer to the display. The pasted display element is an exact copy of the display element in the paste buffer. Only the position is changed. The pasted display element is incremented by 25 pixels in the X and Y direction. A copy of pasted display element remains in the paste buffer so that you can continue to paste the display element in additional locations. The next Cut or Copy command will overwrite the current display element in the paste buffer. |
| Delete | Del | | Delete deletes the current selected display element. The deleted display element is not placed in the paste buffer, and so can not be pasted back. The current display element in the paste buffer is not overwritten. |

View

The View menu provides access to miscellaneous windows, primarily for maintenance type functions, [Table 12](#).

Table 12. View Menu

| Name | Short Cut | Toolbar | Description |
|--------------|-----------|---------|--|
| Element List | | | Display the element list. See Element List on page 97. |

Table 12. View Menu

| Name | Short Cut | Toolbar | Description |
|----------------------|-----------|---------|---|
| Trace Log | | | Display the trace log. See Trace Log on page 268. |
| Communication Status | | | Display the communication status. See Communication Statistics on page 269. |

User

The User menu provides access to dialogs for configuring user preferences and language, [Table 13](#).

Table 13. User Menu

| Name | Short Cut | Toolbar | Description |
|-------------|-----------|---------|--|
| Preferences | | | This displays the User Preferences dialog. Refer to <i>System 800xA Information Management Configuration (3BUF001092*)</i> . |
| Language | | | This displays the Edit Language dialog. Refer to <i>System 800xA Information Management Configuration (3BUF001092*)</i> . |

Change

The Change menu provides functions for aligning, ordering, and further manipulating display elements, [Table 14](#).

Table 14. Change Menu













| Name | Sub-menu Name | Toolbar | Description |
|-------|---------------|---|--|
| Align | Options | | Displays the Align Options dialog. Use this dialog to specify which of the selected display elements, the other display elements should align to. Then use the Align menu to align selected display elements to the Left, Center, Right, Top, Middle or Bottom. To align a number of display elements, first specify the appropriate align option, click on the display elements you want to align, and then choose the appropriate menu item from the Align submenu. |
| | Left |  | Align selected elements to the left according to specified Align Options . |
| | Center |  | Center selected elements according to specified Align Options . |
| | Right |  | Align selected elements to the right according to specified Align Options . |
| | Top |  | Align selected elements to the top according to specified Align Options . |
| | Middle |  | Align selected elements to the middle according to specified Align Options . |
| | Bottom |  | Align selected elements to bottom according to specified Align Options . |
| | Distribute | | Displays the Distributing Display Elements dialog. |

Table 14. Change Menu

| Name | Sub-menu Name | Toolbar | Description |
|---------|---------------|---|---|
| Arrange | Group |  | Group selected elements. A group is a new display element created by combining a number of selected display elements into a single unit. The grouped display element can be resized and moved around like any other display element, and it can even change color dynamically. Refer to Grouping Display Elements on page 94 for further information. |
| | Ungroup |  | This ungroups elements that were previously grouped so they can be treated as individual display elements again. |
| Order | Move Front |  | Move selected element one step forward. |
| | Move Forward |  | Move selected element to the front. |
| | Move backward |  | Move the selected element one step backward. |
| | Move Back |  | Move the selected element to the back. |
| Grid | | | Displays the Grid Settings dialog. The grid is a set of points or lines at specific intervals to help you measure and align the display elements on the display. |

Window

The Window menu provides window handling functions, [Table 15](#).

Table 15. Window Menu

| Name | Short Cut | Toolbar | Description |
|---------------|-----------|---------|--|
| Close | | | Close the current window. |
| Close All | | | Close all open windows. |
| Cascade | | | Display all open windows in cascade fashion. |
| Tile | | | Display all open windows as tiles. |
| Arrange Icons | | | Move all minimized displays to the bottom of the canvas. |

Application Guidelines

Performance

The limit for simultaneous display sessions is 32. Each PC-client is one session.

The recommended total display values per second should not exceed 150 values/sec. This is calculated by adding the data subscriptions for all the concurrent active Display Services windows. Use the worst case (that is to say, the display with the most requests). You should evaluate other processes to determine the overall allowable load.

This product is intended for management information display support, and not as a process control interface. It is recommended that subscription rates be analyzed for each application in order to provide reasonable monitoring capability while minimizing the load on the display server.

Application Hints

- Read these hints to avoid potential pitfalls when building and running displays.
- As you build display, you should periodically test the display in Run mode to determine if the scripts for data access are correct. This will simplify the debugging process as you build your displays.

- In complex displays with many display elements, some elements may be hidden behind other elements. It may be difficult or even impossible to select hidden display elements in order to manipulate them as described above. In such cases use the [Element List](#) to access hidden display elements. You can then perform the above functions directly from the Element List. See [Element List](#) on page 97.
- It is recommended that the PC client has a color palette of more than 256 colors if displaying bitmaps (or other graphic formats) in order to achieve the correct colors.
- Access to process objects is case sensitive.
- On switching from build to run mode, *On Entry* and *On PreQuery* actions do not apply. Use the display function. See [display](#) on page 303.
- To embed quotes in a text string, refer to [Strings](#) on page 281 for proper syntax.
- When choosing fonts, ensure that your PC clients support the chosen fonts.
- When you build displays on a screen with greater resolution than the screen where the displays will ultimately run, you may have to adjust the base size of some fonts to maintain alignment. Refer to [Appendix D, Font Scaling Guidelines](#) for details.
- Grouping inside user elements may result in undesirable affects. Certain elements may not be visible.
- A matrix can not be passed into a user element as an input parameter.

SQL Queries

SQL queries are processed by the ADO data provider on Windows platforms). How to configure this data provider is described in *System 800xA Information Management Configuration (3BUF001092*)*.

All OCS objects as described in the *AdvaInform Object Types Reference Manual* may be accessed with SQL queries to the ADSdpDYN data provider on HP-UX platforms. However, SQL queries to the ADO data provider on Windows cannot be used to access many of these OCS objects. This includes basic objects such as AI, AO, DI, and DO, process objects such as CCF loop and FCM objects, TCL objects, and user objects built with the AdvaBuild Object Type Builder. Nor can SQLPlus be

used to access these objects. When you need to access such objects from a Windows-based server, use an alternative method such as DCS subscription.



SQL access to numeric log data on a Windows-based History server is supported when the EH Data Provider software is installed on the server.

SQL Queries for Numeric Log Data

SQL queries for numeric log data require an ADO data provider that provides Oracle access via the Open Data Access (ODA) database named Database1. The easiest way to create this data provider is to copy the existing ADO data provider, and make the following two changes to the data provider argument list:

- Change the -name argument, for example: from DBA to **DBA1**.
- Set the -dbname argument to **Database1**.

For further details on how to copy and configure data providers, refer to the section on configuring data providers in *System 800xA Information Management Configuration (3BUF001092*)*.

Display Applications Index

XY Plot

This application requires two display elements: [XYPlot](#) and [Matrix](#). To coordinate interaction between the Matrix and XYPlot elements, see [Guidelines for Building an XYPlot](#) on page 133.

Reading Data from a MOD 300 TCL Unit Array

Display Services can read data from a TCL unit array via a [Matrix](#) display element. See [Reading Data From a TCL Unit Array](#) on page 135.

Reading Data from a Trend Display Element

The [Matrix](#) element can be used to extract data from a [Trend](#) for calculations. See [Reading Data from a Trend Display Element](#) on page 137.

Writing to External Applications

Display Services can write to Advant process objects and well as OPC objects. Display services can also add numeric entries to a History log, or modify existing entries. These write transactions are executed by *Data* statements in the [Matrix](#) display element DataQuery property. See:

- [Modifying or Adding an Entry for a Numeric History Log](#) on page 139.
- [Writing to an Advant Process Object](#) on page 139.
- [Reading From or Writing to an OPC Object](#) on page 140.

OPC Data Access

You can query for OPC data by using a data statement in the numeric and text display elements. This is demonstrated in the [Section 3, A Quick Tutorial](#).

Reading the Last History Value Rather than the Real-time Process Value

You can have numeric elements retrieve data from a process object's corresponding History log rather than the real-time value. This reduces the load on the control network, and improves network performance. This functionality is configured via a user preference: AID-DATARETRIEVAL-DCSDATA. Details are provided in *System 800xA Information Management Configuration (3BUF001092*)*. The numeric element is configured the same way, whether you are querying for real-time data, or Historical data.

You can also use the *Data* statement in a Matrix element to subscribe to the last history value. See [Querying for the Last History Value](#) on page 141.

Section 5 Display Services Configuration

Introduction

Displays are configured on two levels. There are procedures and operating parameters related to the display itself. These are described in [Working with Displays](#) on page 73. Then there are procedures and operating parameters related to the individual display elements. These are described in [Working with Display Elements](#) on page 102.

For instructions on configuring custom display elements, refer to [Working with User Elements](#) on page 227.

Other related topics in this section are:

- [Trace Log](#) on page 268
- [Communication Statistics](#) on page 269
- [Run-time Considerations](#) on page 270

Working with Displays

Displays are built by combining display elements such as trend graphs, push-buttons, numeric fields, and so on.

Opening a Display

When you start Display Services in run mode, displays are opened in run mode by default. When you start Display Services in build mode, displays are opened in build mode by default. To open a display, find the display in the [Object Browser](#), then either double-click, or right click and choose **Open** from the context menu.

Adding a New Display

All displays must belong to a display group. A group is a folder on the [Object Browser](#) tree. Before you can create a new display, there must be at least one Group folder on the object browser tree. To add a new group, use the [Add Child](#) or [Add Sibling](#) commands which are available via the [Object Browser](#) context menu. For an example of how to add display groups and displays, see [How to Create a New Display](#) on page 36.

Configuring Display Properties

You can configure display properties to determine size, color, and other aspects of the display's behavior and appearance. You can also configure events to be executed upon entering or leaving a display. Display properties are configured via the [Properties Dialog](#). For details, see [Display Properties](#) on page 87.

Determining the Current Display User for Scripting Applications

The `getpref` function in the Display Services scripting language lets you get the current display user and then set certain operating parameters based on the user. For details regarding the `getpref` function, refer to [getpref](#) on page 307.

Display Elements

Use the [Toolbar](#) to select the display element that you want to insert in a display. For an example, see [How to Insert a Display Element](#) on page 38.

You can define the properties of each display element on an individual basis. Each type of display element has a different set of properties that define its appearance and behavior. For instance, some of the properties of the Shape display element include line width, direction, and shape. The properties for a Button display element include Action, Caption, and Status.

Properties are configured initially in Build mode using interactive dialogs and scripts. See [Working with Display Elements](#) on page 102. Properties can be made to change dynamically in Run mode by executing scripts, either on demand (for example by activating a push button), or based on a pre-determined event. The available functions and syntax rules for scripts are described in [Section 6, Display Scripting](#).

The following functions help you manipulate display elements within a display:

- Cut, copy, paste, and delete via the [Edit](#) menu. See [Main Menu Bar](#) on page 62.
- Use a grid to size and arrange display elements. See [Using a Grid](#) on page 91
- Align display elements. See [Aligning Display Elements](#) on page 92.
- Distribute display elements. See [Distributing Display Elements](#) on page 94.
- Group display elements. See [Grouping Display Elements](#) on page 94.
- Specify the execution order and front-to-back order of display elements. See [Ordering Display Elements](#) on page 95.



In complex displays with many display elements, some elements may be hidden behind other elements. It may be difficult or even impossible to select hidden display elements in order to manipulate them as described above. In such cases use the [Element List](#) to access hidden display elements. You can then perform the above functions directly from the Element List. See [Element List](#) on page 97.

Properties Dialog

The physical appearance and behavior of the display and its individual display elements are specified by configuring properties. You configure the properties of the display and each display element on an individual basis. This is done via the Properties dialog, [Figure 40](#).

Using this dialog, you can view and edit properties for either a display element, or a display. For details on operating this dialog, see [Operating the Properties Dialog](#) on page 76.

Opening the Properties Dialog for a Display

To display the Properties dialog for the display itself, first make sure all display elements are NOT SELECTED, right click anywhere inside the display, then choose **Properties** from the context menu.

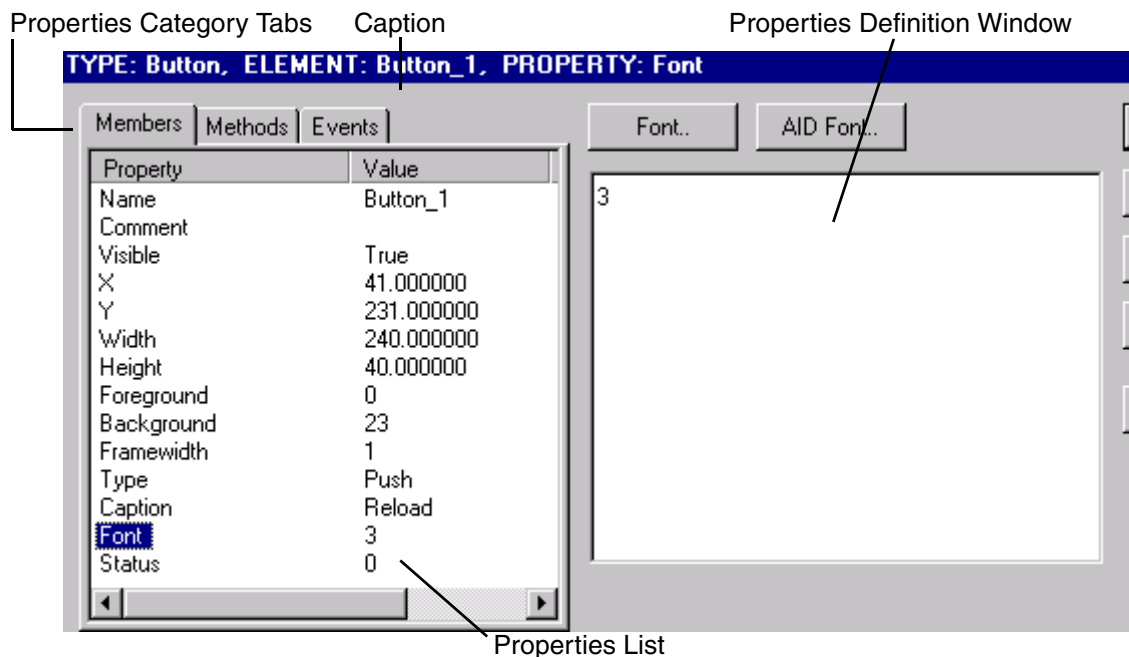


Figure 40. Properties Dialog for Button Display Element

Opening the Properties Dialog for a Display Element

To display the Properties dialog for a specific display element:

1. First click on the display element to select it.
2. Right click on the selected element to display the context menu.
3. Then choose **Properties** from the context menu. The property list in this dialog varies, depending on the kind of display element you selected.

Operating the Properties Dialog

The operation of the Properties dialog is basically the same for all display elements. The actual properties that are available for configuration vary, depending on the type of display element.

To configure a property in the Properties dialog:

1. Click on the applicable tab (**Members**, **Methods**, or **Events**) to show the corresponding properties in the [Properties List](#).
2. Select the property you want to configure from the list.
3. Edit the property definition by one of the following methods:
 - a. For color properties, you can either use the [Color](#) palette, or enter the numeric representation for the color directly in the [Property Definition Window](#).
 - b. For font properties, use the [Font](#) dialog, or enter the numeric representation for the font directly in the [Property Definition Window](#).
 - c. For Menu properties, use the [MenuEditor](#).
 - d. For enumeration properties, choose a value from the [Choices List](#), or enter the value directly in the [Property Definition Window](#).
 - e. For all other data types, you must enter the value (or script) directly in the [Property Definition Window](#).



You can copy the contents of the Property Definition Window, and paste it in the Property Definition Window of another property. See [Property Definition Window](#).

Caption Bar

The caption bar indicates the type of display element (or display), the name, and the currently selected property. For instance, the caption in [Figure 40](#) indicates the display element type is **Button**, the name is **Button_1**, and the currently selected property is **FONT**.

Properties List

The Properties list provides access to all configurable properties for the selected display element (or display). Properties are categorized under three different tabs:

- **Members** - These are variables or constants such as strings and integers.
- **Methods** - These are functions that are called by the environment. Methods are only available for [Working with User Elements](#) and [ActiveX Controls](#).

- Events - These are [Actions](#) (scripts) that execute on certain events, such as when a push button is activated or a field changes.

The property name is indicated in the left column and the current value/definition is indicated in the right column. If the list is too long for the window, a scroll bar is provided.

To configure a property, first select it in this list. The Properties dialog then activates the windows, fields, and buttons as required to configure the selected property. For instance, when you select the FONT property, as shown in [Figure 40](#), the dialog provides two buttons (FONT and AID FONT) which display two different dialogs for changing the font selection.

If the data type of the property is enumeration (meaning there is a predefined list of choices), the choices are indicated in the [Choices List](#) (upper middle portion of the window).

The properties for each type of display element are described in [Working with Display Elements](#) on page 102.

Property Definition Window

Use this window to define the selected property whether the definition is a string, numeric value, or script. For an example of entering a script in the Property Definition window, see [How to Define a Query](#) on page 43.

Some properties such as color and font have associated dialog boxes. For these properties you can either use the dialog box or enter the value (numeric representation) directly in this window.

You can copy the contents of the Property Definition Window, and paste it in the Property Definition Window of another property. To do this:

1. Select the property whose definition you want to copy.
2. Highlight the text to copy either by using the mouse or by using the SHIFT & cursor keys.
3. Press CTRL+C to copy or CTRL+X to cut.
4. Select the property where you want to paste the copied text. The property can either be in the same object (that you copied the text from) or a different one.

5. Place the cursor in the Property Definition Window at the insertion point.
6. Press CTRL+V to paste.

Choices List

The choices list is displayed when you select a property whose value must be chosen from a discrete list. For example, the Type property for a button can be Push or Toggle, [Figure 41](#).

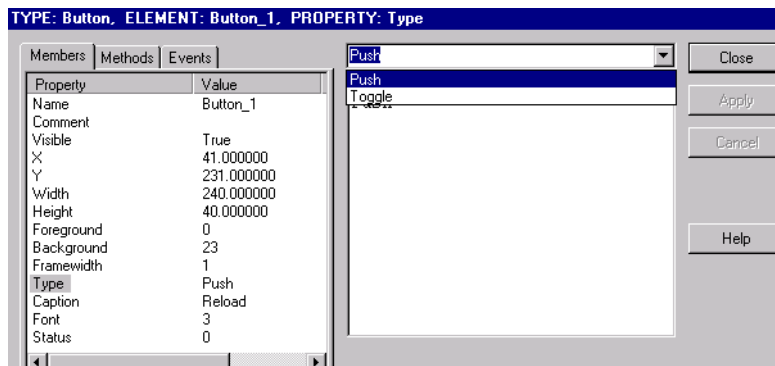


Figure 41. Example, Choices List

To define the property, you can either select a value from this list, or enter the value directly in the [Property Definition Window](#).

Last Check Window

This displays syntax checker messages. See [Figure 21](#) in [How to Define a Query](#) on page 43.

Check Button

This button runs a syntax checker for scripts. Syntax checker messages are displayed in the [Last Check Window](#).

Color

Colors are represented numerically in the Properties dialog. For instance, in [Figure 42](#), the Foreground and Background properties of the display element are defined as 0 and 23 respectively.

When you select a property whose data type is color, the Properties dialog provides two buttons for displaying two different color palettes, [Figure 42](#).

You can either define the color directly by entering the appropriate number in the Property Definition window, or click one of the buttons to display the corresponding palette. Click **Palette** to display the [Windows Color Dialog](#) or **AID Palette** to display the [AID Color Dialog](#).



To specify Transparent (no color), enter **-1** directly in the Property Definition window.

If you need to determine the number corresponding to a certain color, select the color in the Color Palette, and then read the corresponding numeric value in the Properties List of the Properties window.

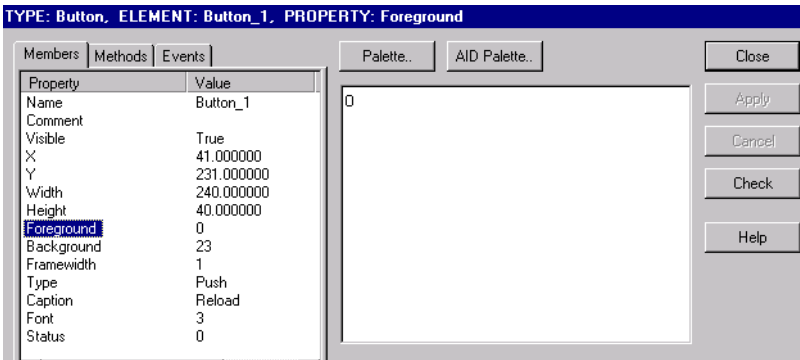


Figure 42. Color Palette Buttons in Properties Dialog

Windows Color Dialog

Use the Windows Color dialog to select colors when you plan to run the display on a PC-client (Windows platform). The [AID Color Dialog](#) supports earlier (non-Windows-based) platforms. The Windows Color dialog is shown in [Figure 43](#). For basic operation, simply select a color, then click **OK**. To use more sophisticated features of this dialog, click the help button (?) for further information.

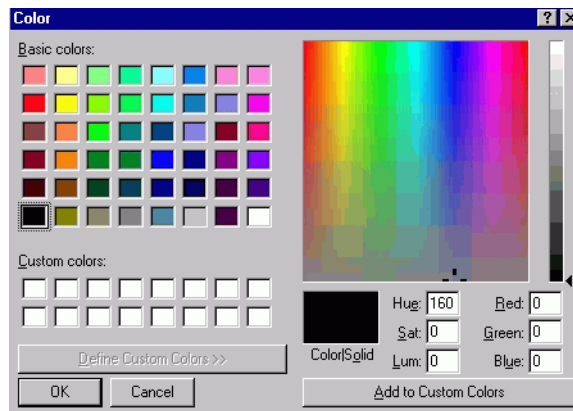


Figure 43. Windows Color Dialog

AID Color Dialog

Use the AID Color dialog, [Figure 44](#), to select colors when you plan to run the display on an X-client (HP platform). If you plan to run the display on an PC-client (Windows platform), use the [Windows Color Dialog](#).

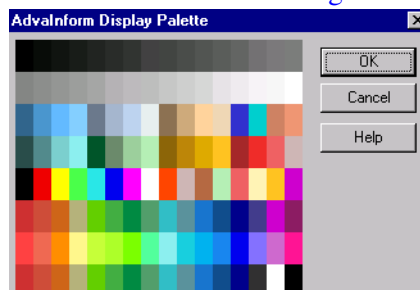


Figure 44. AID Color Palette

The Color palette has 128 buttons, one for each of the 128 colors. The first 124 colors are equal to the first 124 colors (A1-H12) of the Advant Station Operator Station color palette. The last 4 colors are the special colors used in the operator station for printing (they are mapped to other colors before hardcopy).

To change the color of a property, you can either use this palette, or enter the color definition directly in the Property Definition Window. To use this palette, click the color you want, then click **OK**.

Font

Font types are represented numerically in the Properties window. For instance, in [Figure 45](#), the Font property of the button is defined as 3. When you select the Font property, the Properties dialog provides two buttons for displaying two different font selection dialogs, [Figure 45](#).

You can either define the font directly by entering the appropriate number in the Property Definition window, or click one of the buttons to display the corresponding font list. Click **Font** to display the [Windows Font Dialog](#) or **AID Font** to display the [AID Font Dialog](#).

If you need to determine the number corresponding to a certain font, select the font in the Font dialog, and then read the corresponding numeric value in the Properties List of the Properties window.

When choosing fonts, ensure that your PC clients support the chosen fonts.

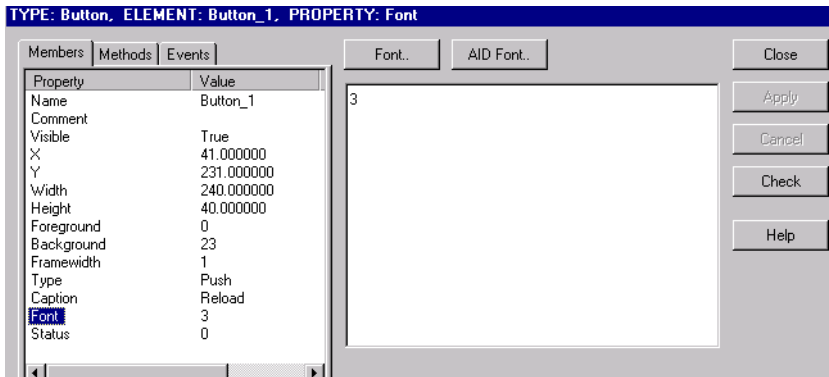


Figure 45. Font Selection Buttons in Properties Dialog

Windows Font Dialog

Use the Windows Font dialog to select fonts when you plan to run the display on a PC-client (Windows platform). If you plan to run the display on an X-client (HP platform), use the [AID Font Dialog](#).

The Windows Font dialog is shown in [Figure 46](#). For basic operation, simply select a font specification (type, style, and size), then click **OK**. To use more sophisticated features of this dialog, click the help button (?) for further information.

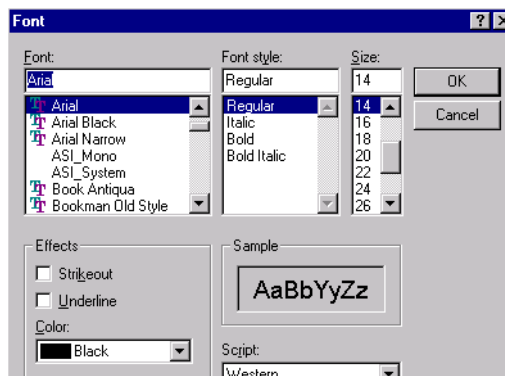


Figure 46. Windows Font Dialog

AID Font Dialog

Use the AID Font dialog, [Figure 47](#), to select fonts when you plan to run the display on an X-client (HP platform). If you plan to run the display on an PC-client (Windows platform), use the [Windows Font Dialog](#).

Use the Font window to select a font type (or family), and the Size window to select the font size. A key to the font types represented by FA through FL is provided in [Table 16](#). When you make a selection in this dialog, a sample is displayed to show the actual selected font.

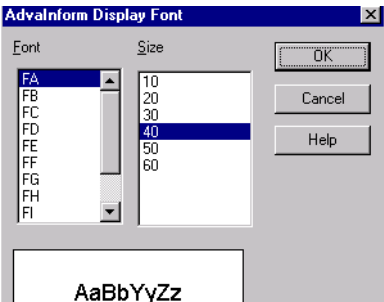


Figure 47. Font Dialog Box

Table 16. Font Types

| Designation | Definition | Designation | Definition |
|-------------|----------------------|-------------|---------------------------|
| FA | Arial, Normal | FH | Time Roman, Bold & Italic |
| FB | Arial, Italic | FI | Courier, Normal |
| FC | Arial, Bold | FJ | Courier, Italic |
| FD | Arial, Italic & Bold | FK | Courier, Bold |
| FE | Times Roman, Normal | FL | Courier, Bold & Italic |
| FF | Times Roman, Italic | | |
| FG | Times Roman, Bold | | |

MenuEditor

The Menu Editor lets you add custom menus and toolbar buttons to the display you are building. Custom menus and toolbar buttons are properties of displays and shown when the display is in run mode.

To open the MenuEditor, [Figure 48](#), select the Menu property in the properties dialog box for a Display.

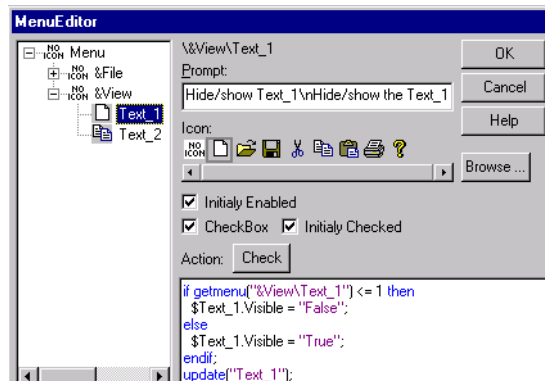


Figure 48. Menu Editor

Menu Tree

The menu tree shows the hierarchy of menu items. From this tree you can select existing menu items, and add new menu items. Use the right click menu.

Menu Item Path

This specifies the placement of the menu item in the hierarchy. For example, \&View\Text_1 specifies **Text_1** as a menu item in the **View** menu.

Prompt

This is the text which is displayed in the Status bar when the menu item is highlighted. You can also specify tool tip text if the menu item is included in the toolbar. For example, in [Figure 48](#) the prompt text is Hide/show Text_1. The text following \n is the tool tip text (in this case, Hide/show Text_1).

Icon

You can associate any one of the icons in this list with the selected menu item, or you may associate no icon with the menu item. The icon is only shown if you choose to include the menu item in the toolbar.

To include a menu item in the toolbar, use any one of the following methods:

- drag the icon from the menu tree or from the icon list to the toolbar
- right click on the menu item and then choose **append to toolbar**.

To remove a menu item from the toolbar, right click on the menu item and then choose **remove from toolbar**.

Browse

The Browse button lets you select another icon file. All icons for the display must be selected from the same icon file, so changing the icon file affects all icons. The icon file must be located in the same file path for all AID display clients that run this display.

Initially Enabled

This checkbox lets you specify whether the menu item (and corresponding icon in the toolbar) are initially enabled or disabled (dimmed).

See the setmenu and getmenu functions in [Section 6, Display Scripting](#) for details regarding menu states.

Check Box

This check box lets you select whether or not to have the menu item operate like a check box. See the setmenu and getmenu functions in [Section 6, Display Scripting](#) for details regarding menu states.

Initially Checked

If the menu item is configured to operate as a check box, this checkbox lets you specify whether the menu item is initially checked (and corresponding icon in the toolbar is initially down).

See the setmenu and getmenu functions in [Section 6, Display Scripting](#) for details regarding menu states.

Action

This is the script to be executed when the menu item is selected. Keywords in the script are color-coded. Pressing the F1 function key on a keyword displays the corresponding help topic.

Menu names and menu items with children can not have actions.

Check

Use this button to check the [Action](#) syntax.

Display Properties

Properties for the display itself are listed in [Table 17](#) and [Table 18](#). Like display elements, you can select the size and position and the background color.

Table 17. Display Member Properties

| Property | Type | Default | Range | Comment |
|--------------------------------|---------|--------------|-------------------|--|
| Description | String | NONE | 49 characters max | Text string for a user-defined description of a display. |
| X | Integer | As created | 0 to 1100 | X-position of upper left corner of the display on your screen. |
| Y | Integer | As created | 0 to 900 | Y-position of upper left corner of the display on your screen. |
| Width | Integer | As created | 0 to 1200 | Width of the display. |
| Height | Integer | As created | 0 to 1000 | Height of the display. |
| BackGround | Integer | 16 | 0 - 127 | The display background color. |
| Menu | String | NONE | | User defined menus. |
| UserProperty | String | NONE | | User Defined properties |
| FuncKeyElement | String | ABBBButton_1 | | The name of the ABBBButton bar for binding the F1-F10 keys |

Table 18. Display Event Properties

| Property | Type | Default | Comment |
|----------------|--------|---------|--|
| OnPreQuery | Action | NONE | Script for an action executed when entering the display before dataqueries of the elements are performed. |
| OnEntry | Action | NONE | Script for an action executed when entering the display after dataqueries of the elements are performed. |
| OnLeave | Action | NONE | Script for an action executed when leaving the display. |
| Broadcast | Action | NONE | Action to be executed when a broadcast is received. |
| ConnectionInfo | Action | NONE | Action to be executed when the status of a connection changes. |

Description

The Description property is a text string for a user-defined description of a display. If a description is defined, the description replaces the display name in the title bar of the display when you put the display in Run mode. The Description can not be modified in Run mode.

OnPreQuery, OnEntry and OnLeave

These properties let you schedule actions to be executed upon entry and exit of the display.

- OnPreQuery

This action is executed when you enter the display from another display in Run mode, **before** the display elements have performed their data queries. The OnPreQuery action is typically used to set up session variables and other initializing.
- OnEntry

This action is executed when you enter the display from another display in Run mode, **after** the display elements have performed their data queries. The OnEntry action is typically used to set up data queries of elements depending on other elements.



OnPreQuery and OnEntry actions are not executed when switching from Build mode to Run mode.

OnLeave This action is executed just before switching to another display in Run mode, or when you exit Display Services. The OnLeave action is typically used to save session variables or set up session variables for the next display.



Do not execute a ReQuery action within an OnPreQuery action.

UserProperty

Selecting **UserProperty** displays the User Property definition dialog. Refer to [Working with User Elements](#) on page 227 for a description of User Properties.

The User Properties of type Member and Event can be initialized in the display(...) call. For instance, consider display 'MY_DISP' in group 'MY_GROUP' with the following User Properties:

STATUS (of type Member - String)

VALUE (of type Member - Integer)

The call to switch to the display from another display would be:

```
display("MY_DISP", "MY_GROUP", "OK", 5023);
```

This causes the display to be loaded, STATUS is initialized to 'OK' and VALUE is initialized to '5023'. In the 'MY_DISP' display, the User Properties can be referenced as:

```
$Display.STATUS
```

```
&Display.VALUE
```

Menu

You can customize the display menu bar and create your own menus. See [MenuEditor](#) on page 84. The Menu property can not be modified in Run mode.

FuncKeyElement

The FuncKeyElement property is the name for the ABBButton display element. FuncKeyElement buttons correspond to the function keys F1 to F10 on the keyboard. When the Display Services window is in focus, pressing a function key

F1 to F10 provides the same functionality as clicking on the corresponding ABBButton.



If you run Display Services in an X-environment, you should press <Ctrl-Fn>.

Broadcast

Broadcast is an action which is executed when the display client receives a broadcast message. The message received is stored in the display's internal *message* property. All clients connected to the display server (including the sender of the broadcast message) receives the message, and it is the contents of the 'Broadcast' property of the display that determines what to do with it.

Example:

- **Sending**

To send the string "TEST" when pushing a button:

The 'Action' property of the button shall contain:
`broadcast("TEST");`

- **Receiving**

To display the received string in a text element (Text_1), the Broadcast property of the display shall contain:

```
$Text_1.DataQuery = $Display.Message;  
update(Text_1);
```

ConnectionInfo

ConnectionInfo provides a method for determining when Data Providers have changed state and allows an appropriate action to be coded. For example:

```
If (&Display.DPstatus = 1) then dialog("Informational","Display Refresh","Data  
Provider Restart " + $Display.DPname);  
else dialog("Informational","Display Refresh","Data Provider Disconnect " +  
$Display.DPname);  
endif;  
display("Test","yourdisplay");
```

Using a Grid

The grid is a set of points or lines at specific intervals to help you measure and align the display elements on the display. Toggling the **Grid** button, [Figure 49](#), alternately displays and hides the grid, according to the [Grid Settings](#).

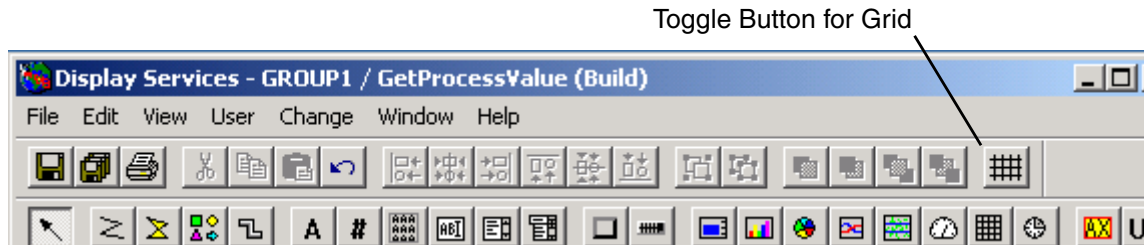


Figure 49. Toggle Button for Grid

Grid Settings

The grid settings specify the appearance, style and behavior of the grid. To specify these settings choose **Change > Grid** from the main menu bar. This displays the Grid Settings dialog, [Figure 50](#).

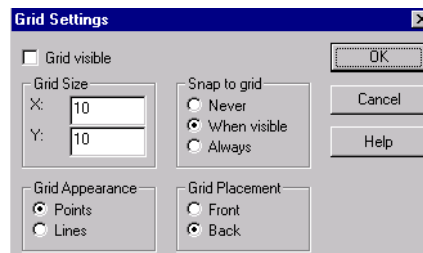


Figure 50. Grid Settings

The options are: [Grid Visible](#), [Grid Size](#), [Grid Appearance](#), [Grid Placement](#), [Snap to Grid](#).

To save the new settings and close the dialog box, click **OK**. When saving your display, the Grid settings are saved for the next time you open the display in Build mode. The grid is not displayed in Run mode. If the grid is visible in Build mode, and you switch to Run mode, the grid will be visible again when you return to Build mode.

Grid Visible

This check box is equivalent to using Grid toggle button on the toolbar. Use this to determine whether or not to show the grid.

Grid Size

Use this to set the x and y intervals for the grid (space between horizontal and vertical lines or dots). The range is 2 to 50. Default is X=10 and Y=10.

Grid Appearance

Use these radio buttons to select solid lines or dots.

Grid Placement

Use these radio buttons to specify whether the grid should appear in front of all elements (Front) or behind all elements (Back).

Snap to Grid

Use Snap to Grid to specify that all display element operations such as drawing and positioning will be aligned to the grid. The alignment will affect both the position and size. There are three options:

- **Never** - Do not snap to grid whether the grid is visible or not.
- **When visible** - snap to grid only when the grid is visible.
- **Always** - Always snap to grid, even when the grid is not visible.

Aligning Display Elements

The **Align** menu item in the [Change](#) menu lets you align display elements. To align two or more display elements:

1. Choose **Change > Align > Options** from the main menu bar to display the [Align Options](#) dialog. Use this dialog to specify the display element to be used as the point-of-reference by which to align the other display elements.
2. Select the display elements that you want to align.

3. Select the alignment direction. Choose the appropriate menu item from the **Change** menu: **Align>Left**, **Align>Center**, **Align>Right**, **Align>Top**, **Align>Middle** or **Align>Bottom**.

An example is provided in [Figure 51](#). Four text display elements are selected for top-alignment, and the alignment option is First Selected. Since Text_1 is the first display element selected, the other three display elements will be aligned to the top of Text_1.

Align = Tops
Align Option = First Selected

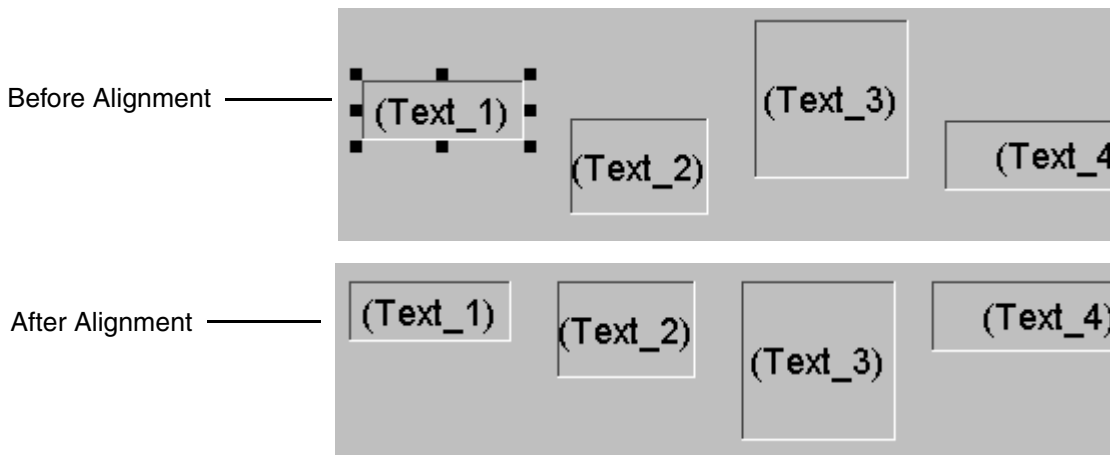


Figure 51. Example, Aligning Display Elements

Align Options

Choose **Change > Align Options** from main menu bar, to specify the display element to be used as the point-of-reference by which to align the other display elements. This displays the Alignment Options dialog, [Figure 52](#). There are three alignment options:

- **First selected:** selected display elements are aligned to the first selected display element.
- **Last selected:** selected display elements are aligned to the last selected.

- **Extreme:** selected display elements are aligned to the left most, top most, right most, or bottom most display element selected (depending on the selected alignment direction).

To select one of the three options, click on the corresponding button. Click **OK** to set the new option and close the dialog box, or **Cancel** to cancel the dialog.

For Center and Middle alignment a geometric center is calculated and used as the alignment reference.

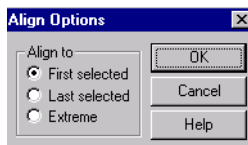


Figure 52. Align Options Dialog Box

Distributing Display Elements

To distribute display elements horizontally and vertically:

1. Select the display elements that you want to distribute.
2. Choose **Change > Distribute** from the main menu bar.
3. Select the appropriate distribution options from the Distribute dialog, [Figure 53](#).

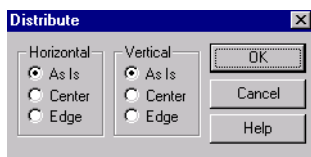


Figure 53. Distribute Dialog

Grouping Display Elements

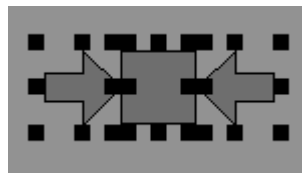
Several display elements can be grouped into a single display element. By grouping several display elements you can change attributes, such as size and visibility, for all grouped display elements at the same time. In Build mode, you can resize or move all display elements within a group as if the group was a single display element.

When you open the Properties dialog for a Group display element, the window contains the common properties for all the individual display elements that comprise the Group. You can edit any property in this dialog to set that property for all individual elements in the Group.

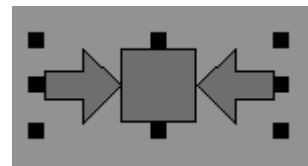
To group display elements, select the display elements to be grouped by holding down the SHIFT key while clicking on the display elements. Then choose **Change > Arrange > Group** from the main menu bar.

To ungroup display elements, select the display element to be ungrouped, and then choose **Change > Arrange > Ungroup**.

How grouped and ungrouped display elements are represented in the Build mode is shown in [Figure 54](#).



Three display elements ungrouped



The same display elements grouped into a single display element

Figure 54. Grouping and Ungrouping Example

Ordering Display Elements

Display elements are processed in a sequence according to their Z-order. Z-order is initially determined by the order in which you add the display elements to the display. The first element added is the first element processed. Z-order also determines which display element will hide another display element when the elements overlap. The last element added is the closest to the front and will hide any element that it overlaps.

You can specify the front-to-back order for display elements by choosing **Change > Order** and then the appropriate order option from the main menu bar, or you can use the equivalent buttons on the toolbar, [Figure 55](#).

- Move Front - This moves the selected display element in front of all other display elements. This display element will be the LAST element processed when the display runs.
- Move Forward - This moves the selected display element one step forward.
- Move Backward - This moves the selected display element one step backward.
- Move Back - This moves the selected display element in back of all other display elements. This element will be the FIRST element processed when the display runs.

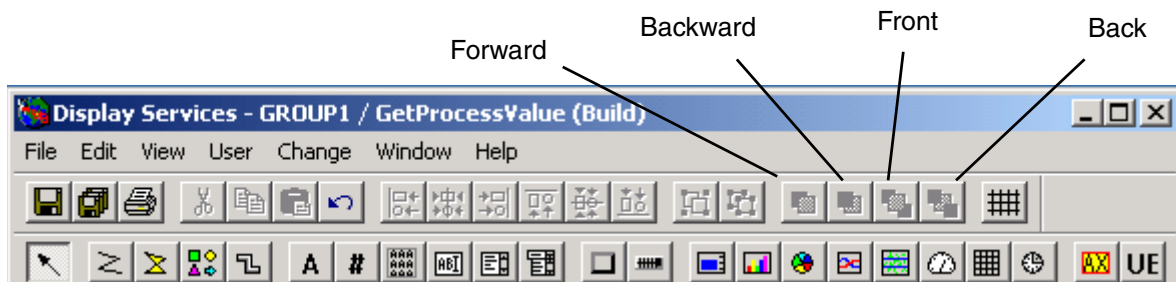


Figure 55. Order Buttons on Toolbar

Some examples are shown in [Figure 56](#).

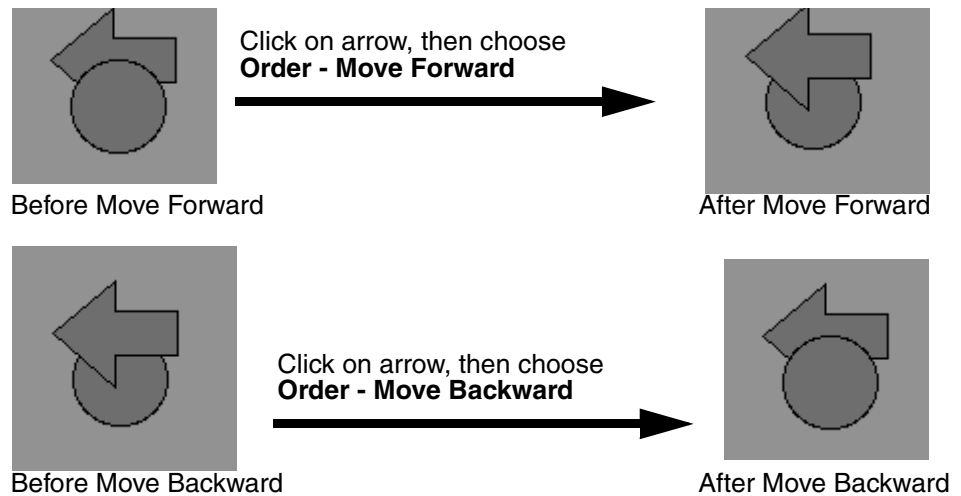


Figure 56. Moving Forward and Backward

Element List

The element list indicates all display elements in the current display, [Figure 57](#). Selecting an element in the element list causes the same element to be selected in the display, and vice versa. This is very useful when you modify the properties of an element that is completely overlapped and thus hidden by another element.

To display the element list for a display, choose **View > Element List** from the main menu bar.

You can use the element list to:

- Select a display element that is hidden behind other display elements on a display, [Selecting an Element in the List](#) on page 99.
- Modify the execution order (Z-order) of display elements, [Changing the Execution Order of Display Elements](#) on page 100.
- Rename display elements, [Renaming a Display Element](#) on page 101
- Delete display elements, [Deleting a Display Element](#) on page 101

The element list has the following fields and buttons: [Name](#), [Type](#), [Z-Order](#), [Order type](#), [Element Count](#), [Selected](#).

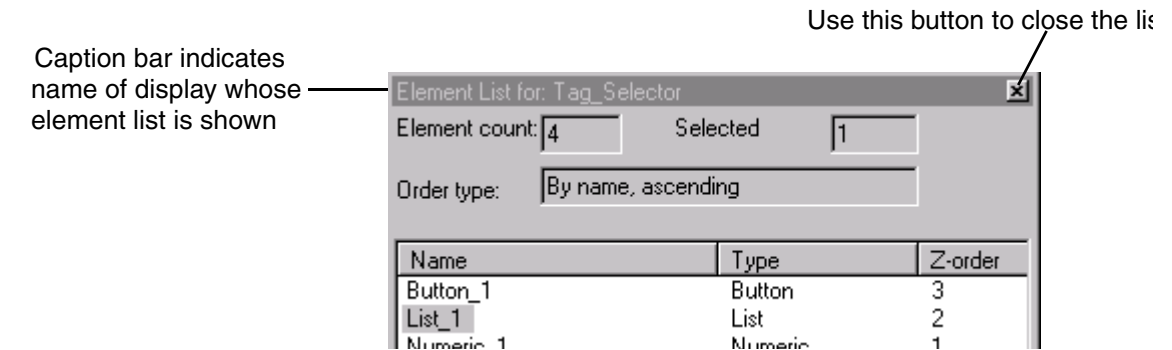


Figure 57. Element List

Name

This column displays the names of the elements. The currently selected element is highlighted. Clicking the **Name** button selects name as the sorting criteria.

Type

This column indicates the type of the element. Clicking the **Type** button selects type as the sorting criteria.

Z-Order

Each display element in a display is assigned a number to represent its z-order. Z-order determines the order in which display elements are processed, and it also determines the front-to-back order for display elements that overlap. Z-order is initially determined by the order in which you add the display elements to the display.

0 (zero) indicates the top of the Z-order. The display element with the z-order value of 0 is the processed first, and is the farthest back. The highest number in the Z-order indicates the display element that is closest to the front. This display element will hide any element that it overlaps, and is the last to be processed.



ComboBox, Edit and List element types are based on standard windows controls. therefore, they are always in front of other element types, regardless of their position in the Z-order. The only way to place an element of another type in front of a ComboBox, Edit or List is to set the Visible property to false.

Order type

This determines how elements are sorted in the list. The choices are

- **By name, ascending:** Sort by names in alphabetical order.
- **By name, descending:** Sort by names in reverse alphabetical order.
- **By type, ascending:** Sort by element type in alphabetical order.
- **By type, descending:** Sort by element type in reverse alphabetical order.
- **Z-order Back -> Front:** Sort by the Z-order of the elements, from back to front. This means elements that overlap other elements are closer to the bottom of the list.
- **Z-order-Front -> Back:** Sort by reverse Z-order of the elements, from front to back. This means elements that overlap other elements are closer to the top of the list.

To select an order type, click on the corresponding button: **Name**, **Type**, or **Z-Order**. Clicking **Name** or **Type** repeatedly toggles between ascending and descending. Clicking **Z-order** repeatedly toggles between Back-to-Front and Front-to-Back.

Element Count

This field displays the total number of elements in the display.

Selected

This field indicates the total number of elements that are selected.

Selecting an Element in the List

To select a single element, click on the element name. To select multiple contiguous elements, select the first element, hold down the SHIFT key, and then select the last

element. To select multiple non-contiguous elements, hold down the CTRL key while you select elements.

Selecting an element in the list also selects the element in the display and visa versa, [Figure 58](#).

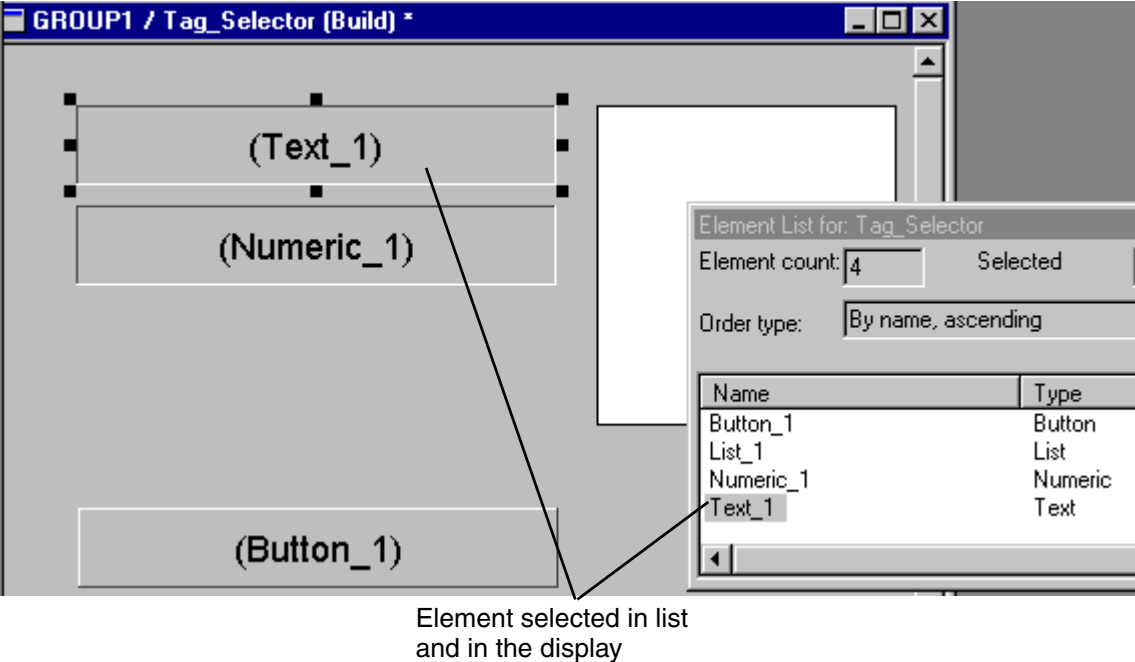


Figure 58. Selecting an Element

Changing the Execution Order of Display Elements

Display elements are processed in a sequence according to their Z-order. Z-order is initially determined by the order in which you add the display elements to the display. The first element added is the first element processed. Z-order also determines which display element will hide another display element when the elements overlap. The last element added is the front most and will hide any element that it overlaps.

You can modify the execution order of elements by using the context menu to move the elements up and down in the list. When more than one element is selected, elements are moved in the same order as they were selected (first selected, first moved). To change the execution order of a display element in the list, right click on an element, and then choose one of the following options from the context menu:

Move Front

The selected element is moved to the top of the list. If sorting method is Back->Front, the element is placed behind all other elements. If sorting method is Front->Back, the element is placed in front of all other elements.

Move Forward

The selected element swaps Z-order position with the element just before it.

Move Backward

The selected element swaps Z-order position with the element just after it.

Move Back

The selected element is moved to the end of the list. If sorting method is Front->Back, the element is placed behind all other elements. If sorting method is Back->Front, the element is placed in front of all other elements.

Renaming a Display Element

To rename a display element, select the display element from the list, right click, and then choose **Rename** from the context menu.

Deleting a Display Element

To delete a display element, select the display element from the list, right click, and then choose **Delete** from the context menu.



This deletes the selected element with no opportunity to confirm.

Working with Display Elements

Display elements are the functional components in displays. Select Display elements to add to a display via the [Toolbar](#). An example of adding a display element is provided in [How to Insert a Display Element](#) on page 38.

The behavior and appearance of display elements are set via the [Properties Dialog](#). An example of configuring display element properties is provided in [How to Configure Display Element Properties](#) on page 40.

A complete list of display element types is provided in [Table 19](#).

Table 19. Display Elements

| Display Elements | | | |
|------------------|----------------------------|------------|--------|
| Timer | Polyline | Polygon | Shape |
| Text | Numeric | MultiText | List |
| Matrix | Edit | Combobox | Button |
| ABBButton | Bar | Multibar | Pie |
| Trend | Gauge | SmartShade | XYPlot |
| ActiveX Control | Working with User Elements | Displays | Group |



You should know how to read and interpret scripts to follow the examples in this section.

You can manipulate display elements in the following ways:

- cut, copy, paste, delete via the [Edit](#) menu. See [Main Menu Bar](#) on page 62.
- align display elements. See [Aligning Display Elements](#) on page 92.
- distribute display elements. See [Distributing Display Elements](#) on page 94.
- group display elements. See [Grouping Display Elements](#) on page 94.
- specify the execution order and front-to-back order of display elements. See [Ordering Display Elements](#) on page 95.



In complex displays with many display elements, some elements may be hidden behind other elements. It may be difficult or even impossible to select hidden display elements in order to manipulate it as described above. In such cases use the [Element List](#) to access hidden display elements. You can then perform the above functions directly from the Element List. See [Element List](#) on page 97.

Timer

The toolbar button for the Timer display element is shown in [Figure 59](#).

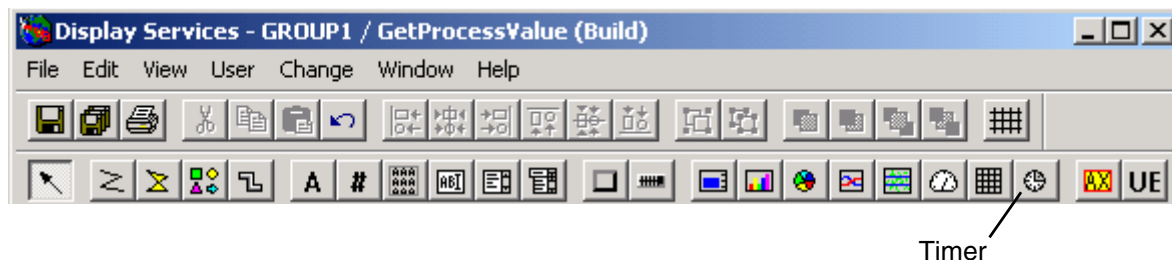


Figure 59. Timer Button in Toolbar

The Timer display element is used to execute an action after a specified time delay, or to cyclically execute an action. Configure the [Timer properties](#), [Table 20](#), to set behavior and appearance characteristics. The Timer display element is not visible in Run mode. You can set the colors and other properties to help you locate Timer elements in Build mode. The Build Mode view of Timer element is shown in [Figure 60](#).

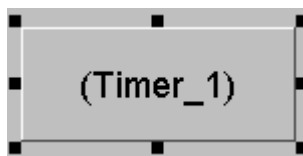


Figure 60. Timer element type

Table 20. Timer Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|---------------------|--|
| Name | String | Timer_# | | Name must be unique within the display. |
| X | Integer | As created | 0 to 1200 | X coordinate of the upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of the upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of Timer in build mode. |
| Height | Integer | As created | 5 to 1200 | Height of Timer in build mode. |
| Foreground | Color | 0 | -1 to 127 | Color of the text. |
| Background | Color | 16 | -1 to 127 | Color of the background. |
| Framewidth | Integer | 3 | 1 to Width/2 | Width of the Motif frame. |
| Mode | Enum | Contiguous | OneShot, Contiguous | The operating mode of the timer. |
| Delay | Integer | 0 | >= 0 | Delay in seconds between executing the Action. |
| Action | Action | | | Action to execute after <i>Delay</i> seconds. |

The following sections provide guidelines for configuring Timer properties. Refer also to [Basic Properties](#) on page 218.

Mode

This property controls the operating mode of the Timer element:

OneShot In this mode, the Timer element executes the action specified in the Action property, after the display is loaded. The delay is specified in seconds by the Delay property. This applies only when switching displays in Run mode, not when switching from Build mode to Run mode. If Delay equals 0, the Action is executed when the display is loaded.

Continuous In this mode, the Timer element executes the action specified in the Action property cyclically at intervals specified in seconds by the Delay property. If Delay is set to 0 seconds, the Action will never be executed.

Delay

This property depends on how Mode is defined. If Mode is OneShot, Delay is the number of seconds until the Action is executed, after the display is loaded. If Mode is Continuous, Delay is the number of seconds between the executions of Action. The number must be greater than or equal to 0.

Action

This is the Action which is executed after a delay or cyclically, depending on the Mode property. For further information see [Action](#) on page 224.

Polyline

The toolbar button for the Polyline display element is shown in [Figure 61](#).

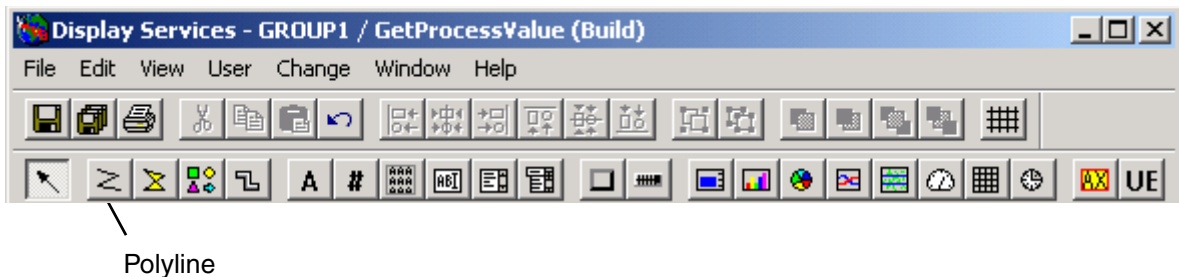


Figure 61. Polyline Button in Toolbar

The Polyline is a component element for building complex symbols, [Figure 62](#).



Figure 62. Example, Polyline

When you draw a polyline, click the left mouse button to end each segment. Click the right mouse button to end the polyline. Configure [Polyline properties](#), [Table 21](#), to set behavior and appearance characteristics.

Table 21. Polyline Properties

| Property | Type | Default | Range | Comment |
|----------------------------|-----------------------|------------|------------------|---|
| Name | String | Polyline_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not line is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of polyline frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of polyline frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of polyline frame. |
| Height | Integer | As created | 5 to 1200 | Height of polyline frame. |
| Foreground | Color | 0 | -1 to 127 | Pen (outline) color of the polyline. |
| Background | Color | 16 | -1 to 127 | Fill color of the polyline. |
| Linewidth | Integer | 1 | 0 to Width/2 | Pen line width. Max = 5. |
| Shape | Enum | Polyline | Polyline,Polygon | Basic shape of the display element. |

Polygon

The toolbar button for the Polygon display element is shown in [Figure 63](#).

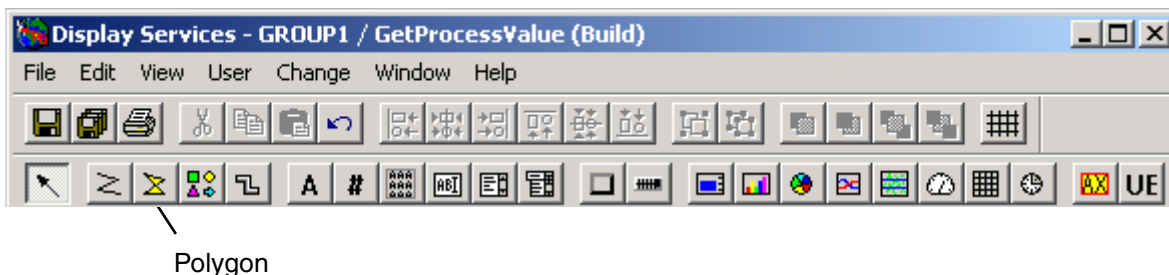


Figure 63. Polygon Button on Toolbar

The Polygon is used as a component element for building complex symbols, [Figure 64](#).

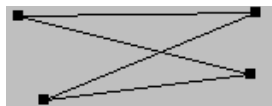
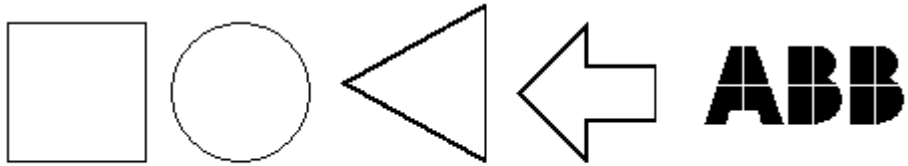


Figure 64. Example. Polygon

When you draw a polygon, left-mouse click to end each segment. Right-mouse click to end the polygon. Configure [Polygon properties](#), [Table 22](#), to set behavior and appearance characteristics.

Table 22. Polygon Properties

| Property | Type | Default | Range | Comment |
|-------------------------|---------|------------|------------|--|
| Name | String | Polygon_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not polygon is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of polygon frame's upper left corner. |



Basic Shapes (BitMap not shown)

Figure 66. Basic Shapes

More examples of Shape display elements are provided on the Demoobject_2 display in the Displaydemo Group. For instructions on how to use demo displays, see [Appendix A, A Demonstration of Display Services](#).

Configure [Shape properties](#), [Table 23](#), to set behavior and appearance characteristics. You can change the position and shape of these shapes either by clicking and dragging, or by re-defining the applicable shape properties. Some shape properties can be made to change appearance by commands from other display elements based on events in Run mode. This is done via scripts.

Table 23. Shape Properties

| Property | Type | Default | Range | Comment |
|-------------------------|---------|------------|------------|--|
| Name | String | Shape_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the shape is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of shape frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of shape frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of shape frame. |

Table 23. Shape Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--|--|
| Height | Integer | As created | 5 to 1200 | Height of shape frame. |
| Foreground | Color | 0 | -1 to 127 | Pen (outline) color of the shape. |
| Background | Color | 16 | -1 to 127 | Fill color of the shape. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |
| Linewidth | Integer | 1 | 0 to Width/2 | Pen line width. Max = 5. |
| Direction | Enum | Left | Left,Up,Right,Down | Direction of the triangle and the arrow. No effect on box and oval. |
| Shape | Enum | Box | Oval, Logo Triangle, Arrow, BitMap, Polyline, Polygon, Pieslice | Basic shape of the display element. |
| BitmapPath | String | | 255 chars | Name of the bitmap file. The name must be entered in double quotes. For example: "test.bmp" |

The property list in the properties dialog changes, depending on the shape type. All the properties can be changed in both Build mode and in Run mode.

Example: Changing the shape and direction of a Shape from a Button

```
$Shape_1.Direction = "Left";
$Shape_1.Shape = "Arrow";
update("Shape_1");
```

BitmapPath

This property specifies the name of the bitmap to import for this shape display element. All bitmap files must reside in a dedicated directory. The default directory on a PC client is C:\Program Files\ABB Industrial IT\Inform IT\Display Services\Client\Bitmap. The bitmap name must be entered in double-quotes, for example:

“test.bmp”

For best performance, bitmaps should be stored on each display client node.

You can change the directory name; however, you must insure that all bitmaps are stored in the same directory. The supported formats are listed in [Table 24](#).

Table 24. Supported Bitmap Formats

| | | | |
|-----------|--------|------------|----------------------------------|
| TIFF | PCD | CUT | PPM |
| JPEG | IGF | DIB | PSD |
| PCX | ICO | IMG (GEM) | RAS |
| TGA | MO:DCA | IMG(Xerox) | SGI |
| PNG | WMF | IMT | WPG |
| DCX | PCT | KFX | XBM |
| JFIF | EPS | RLE | XPM |
| Group3 | | MAC | XWD |
| Group4 | ATT | MSP | Optional formats: ABIC, DICOM |
| Group3 2D | BMP | NCR | |
| CALS | BRK | PBM | |
| IFF | CLP | PGM | |
| IOCA | LV | SUN | |
| ASCII | GX2 | PNM | |

Text

The toolbar button for the Text display element is shown in [Figure 67](#).

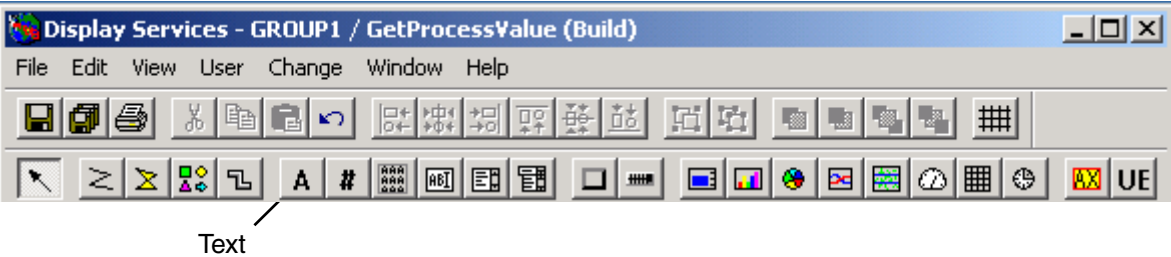


Figure 67. Text Button in Toolbar

The Text display element can be used as a static label, or to retrieve a dynamic text string based on a query. You can also configure an action to execute when the display element is clicked in Run mode. Configure the [Text properties](#), [Table 25](#), to set behavior and appearance characteristics. The text can be aligned vertically (Top, Middle, Bottom) and horizontally (Left, Center, Right). You can select a font from the font list.

To get text data from an OPC object, you must use the data statement rather than a dcsub statement. For details regarding the data statement, see [data](#) on page 295.

The text display element can only have one line of text. If the text string is wider or taller than the display element, the text string is automatically clipped, so it does not exceed the borders of the element. Some examples are shown in [Figure 68](#).

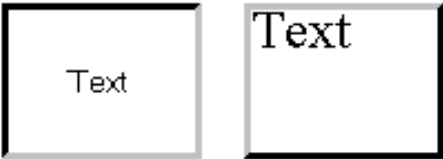


Figure 68. Examples, Text Display Element

Table 25. Text Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--|---|
| Name | String | Text_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the text is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of text frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of text frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the text frame. |
| Height | Integer | As created | 5 to 1200 | Height of the text frame. |
| Foreground | Color | 0 | -1 to 127 | Color of text. |
| Background | Color | 16 | -1 to 127 | Background color of the text (fill color of the frame). |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |

Table 25. Text Properties

| Property | Type | Default | Range | Comment |
|-----------------------------|---------|---------|--------------------|--|
| DataQuery | Query | | | Script for data retrieval. |
| Value | String | | | Value returned by DataQuery (Read only). |
| Vertalign | Enum | Middle | Top,Middle, Bottom | Vertical alignment of the text. |
| Horizalign | Enum | Center | Left,Center, Right | Horizontal alignment of the text. |
| Font | Font | 2 | 0 to 71 | Font number. |
| DataChanged | Action | | | Script for action to execute when Value changes. |
| DataStatus | Integer | 0 | 0,1,2 | Read Only value indicating the status of the DataQuery: 0: OK 1: Pending (green cross) 2: Error (red cross) |
| Action | Action | | | Script for action to execute when you click on the Text element in Run mode. |

The following sections provide guidelines for configuring Text properties. Refer also to [Basic Properties](#) on page 218.

DataQuery

The DataQuery can either be an SQL statement that retrieves a dynamic text string, or it can be a static user-defined text string.

Example SQL statement (DataQuery must be of type string):

```
$sql ("SELECT abc FROM dual")
```

Also see [How to Define a Query](#) on page 43.

Examples, static text string:

```

    "This " + "is " + "possible"
    This_is_a_string,
    "A string with spaces"
    "123"

```

The DataQuery is executed according to the rate specified by the [Update](#) property. If Update mode is specified as Demand, the DataQuery is executed only when the Text display element is updated from another display element (via ReQuery).

Value

The Value property is read-only. It contains the parsed result of the DataQuery, and is normally read by other display elements that require the value. When the DataQuery returns a result that differs from the current Value, the action specified in the DataChanged property is executed.

Numeric

The toolbar button for the Numeric display element is shown in [Figure 69](#).

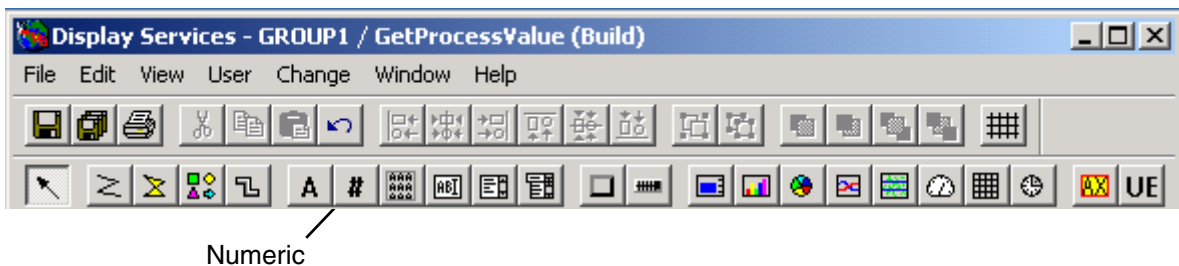


Figure 69. Numeric Button in Toolbar

The Numeric display element is used to retrieve dynamic data values or specify static data values that are numeric (floating point or integer). You can also configure an action to execute when the display element is clicked in Run mode.

Configure [Numeric properties](#), [Table 26](#), to set behavior and appearance characteristics. The numeric values can be aligned vertically (Top, Middle, Bottom) and horizontally (Left, Center, Right).

Numeric elements that collect process data may subscribe to the realtime process value directly from the process object, or the last history value from the

corresponding History log. How to configure this is described in [Subscribing to the Last History Value](#) on page 120.

To get numeric data from an OPC object, for example a softpoint or AC 800M controller object, you must use the data statement rather than a dcsub statement. This is demonstrated in [Section 3, A Quick Tutorial](#).

Other examples of numeric display elements are provided on the Demoobject_1 display in the Displaydemo Group. For instructions on how to use demo displays, see [Appendix A, A Demonstration of Display Services](#).

Table 26. Numeric Properties

| Property | Type | Default | Range | Comment |
|----------------------------|-----------------------|---------------|--------------|--|
| Name | String | Numeric _# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not numeric value is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of the numeric frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of the numeric frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the numeric frame. |
| Height | Integer | As created | 5 to 1200 | Height of the numeric frame. |
| Foreground | Color | 0 | -1 to 127 | Color of the numeric value. |
| Background | Color | 16 | -1 to 127 | Fill color of the numeric frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |

Table 26. Numeric Properties

| Property | Type | Default | Range | Comment |
|-------------|----------|---------|--|--|
| Value | Float | 0 | | Result of the DataQuery (Read only) |
| Vertalign | Enum | Middle | Top,Middle, Bottom | Vertical alignment of values. |
| Horizalign | Enum | Center | Left,Center, Right | Horizontal alignment of values. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| Font | Font | 2 | 0 to 71 | Font number. |
| DataChanged | Action | | | Script for action to execute when Value changes. See DataChanged on page 226 |
| Format | Textline | %.2f | See Format on page 118 | Format of numeric presentation |
| DataQuery | Query | | | Script for data retrieval. See DataQuery on page 222. Examples: 24 56.789 #sql("select 24.68 from dual") &sql("select 24 from dual") 23 + 45 - 67 |

Table 26. Numeric Properties

| Property | Type | Default | Range | Comment |
|------------|---------|---------|-------|--|
| DataStatus | Integer | 2 | 0,1,2 | Read Only value indicating the status of the DataQuery: 0: OK 1: Pending (green cross) 2: Error (red cross) |
| Action | Action | | | Script for action to execute when you click on the element in Run mode. |

The following sections provide guidelines for configuring Numeric properties. Refer also to [Basic Properties](#) on page 218.

Format

The Format property lets you configure the data presentation format for individual numeric display elements. The syntax is as follows:

% [flags] [field-width] [.precision] <conversion character>

The format statement must always start with a percent sign. To mark a numeric value with a percent sign, you must use two percent signs (%%). The other components of the format definition are described below:

[flags]

- - (minus sign) - Left justifies data in the specified output field. Normally data is right justified.
- + (plus sign) - Prefaces data with a sign (either + or -). By default, only negative values are shown with a sign.
- (space) - Inserts a space character before a positive value, when the + flag is not used. This will assure that positive and negative values are aligned when the + flag is not used. By using the space, the first digit of the positive value is aligned with the first digit of a negative value rather than being aligned with the negative sign. If the + flag is set, the space flag is ignored.
- # (hatch) - An alternate form. Refer to [<conversion character>](#) for details.

[field-width]

The field width specifies the minimum number of spaces allocated to the output field. For numeric data; the range is 0-9.

If the data is smaller than the specified field width, the extra space is filled with blank spaces. If the data is larger than the specified field width, the width is expanded to fit the data. If no field width is specified, the resulting field is made just large enough to hold the data.

[.precision]

Precision indicates how many decimal places to show, depending on the conversion character. The syntax is a decimal point followed by a number in the range of 0 - 9. The default is 6. Refer to [<conversion character>](#) for details.

<conversion character>

The conversion character specifies one of the following formats for the data value:

- **f** = decimal notation: *[-]ddd.ddd*

The value is shown in decimal notation, where the number of digits after the decimal point is equal to the value set by `[.precision]`. If no precision is specified, six digits are shown after the decimal point. If the precision is set to zero, the decimal point is eliminated entirely. If the `#` flag is specified, a decimal point is always present, even if no digits follow the decimal point.

- **e** = scientific notation: *[-]d.ddde+-ddd*

The value is shown in scientific notation. There is always one digit before the decimal point. The number of digits after the decimal point is equal to the value set by `[.precision]`. If no precision is specified, six digits are shown after the decimal point. If the precision is set to zero, the decimal point is eliminated entirely. If the `#` flag is specified, a decimal point is used, even if no digits follow the decimal point.

- **E** = scientific notation: *[-]d.dddE+-ddd*

This is the same as **e**, except a capital **E** is used in place of the lower case **e** in the value.

- **g** = decimal or scientific notation

The value is shown as either **f** (decimal notation) or **e** (scientific notation), depending on the size of the exponent. If the exponent is less than -4 or greater than the precision, the **e** format is used. Otherwise, **f** format is used.

The precision specifies the number of significant digits to show. Trailing zeros not shown, and a decimal point is only shown if it is followed by a digit. If the **#** flag is specified, a decimal point will always be used, even if no digits follow the decimal point, and trailing zeros are not removed.

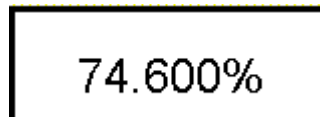
- **G** decimal or scientific notation

This is the same as **g** above, except that **E** is used instead of **e** if scientific notation is used.

The default format of the numeric display element is `%.2f`.

Example:

74.6 with Format=`%.3f%%` is shown in [Figure 70](#).



74.600%

Figure 70. Example, Numeric Display Element

Updating the Numeric data element can be either cyclic or event driven.

Subscribing to the Last History Value

Numeric elements that collect process data may subscribe to the realtime process value directly from the process object, or the last history value from the corresponding History log.



To subscribe to the last history value, the corresponding data provider must reside on the same node where the history log is located. This limitation only applies to last history value, and DOES NOT apply to other historical data queries. For details regarding data providers, refer to the section on data providers in *System 800xA Information Management Configuration (3BUF001092*)*.

The subscription method is specified on a global basis for all numeric elements by configuring the DATARETRIEVAL user preference. The default is for realtime

process data. If you need to change this setting, refer to [Configuring the User Preference for Last History Value](#) on page 121.

If you configure this user preference for realtime data, you can still query for the last History value on a selective (demand) basis. This is done via the data statement in the Dataquery property of a [Matrix](#) display element. The dataquery for the Numeric element must reference the Matrix display element where the last history value is stored.

For details on how to specify the data query for the Matrix element, and how to access data from the Matrix element, see [Matrix](#) on page 131.

For details regarding Data statement syntax, see [data](#) on page 295.

Configuring the User Preference for Last History Value

The default setting for process data retrieval is to subscribe to real-time data. To subscribe to the last history value on a global basis, change the DATARETRIEVAL - DCSDATA preference from **NORMAL** to **HISTORY**, [Figure 71](#). Any changes you make to user preferences will not take affect until you restart the client.

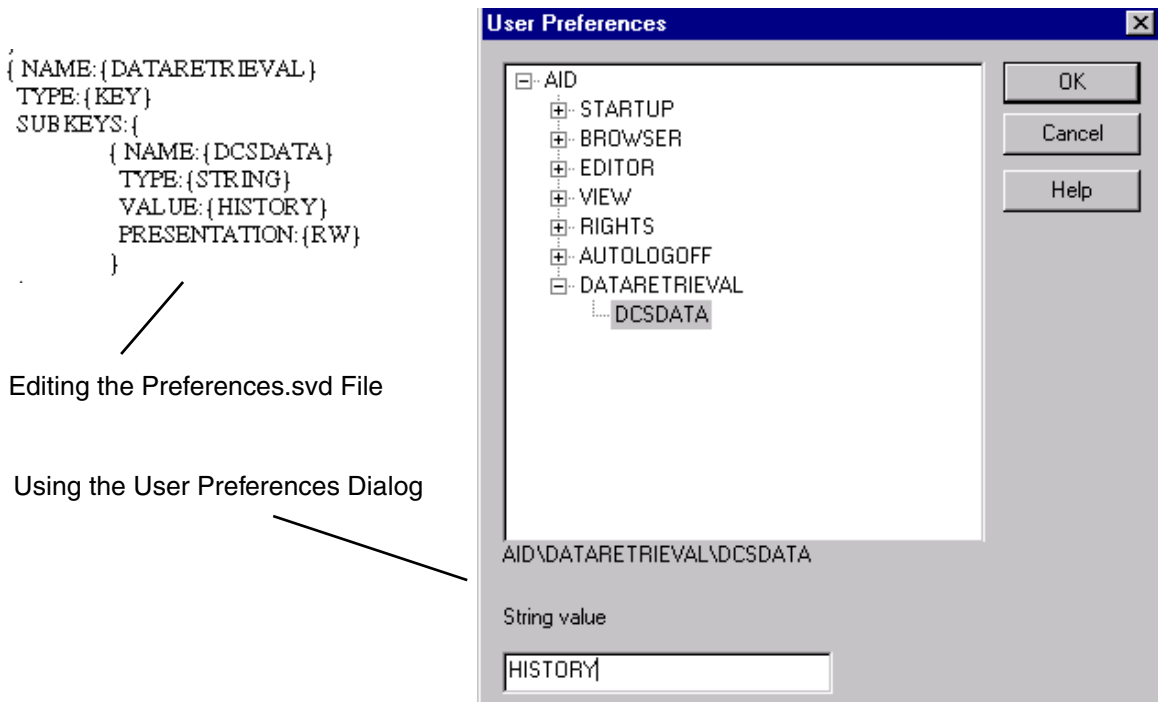


Figure 71. Setting the DATARETRIEVAL - DCSDATA User Preference

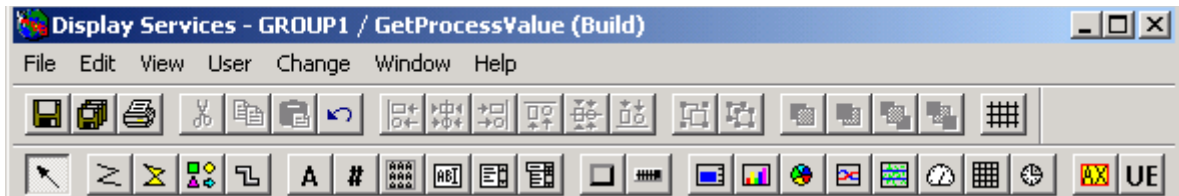


- Whether you configure this user preference for NORMAL or HISTORY, be sure to specify the process object, and not the history object in the dataquery of the Numeric display element. See [DataQuery](#) on page 222.

For further details on configuring user preferences, refer to *System 800xA Information Management Configuration (3BUF001092*)*.

MultiText

The toolbar button for the Multitext display element is shown in [Figure 72](#).

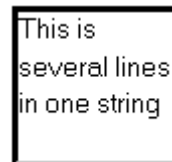


MultiText

Figure 72. MultiText Button in Toolbar

The MultiText display element can display several lines of text, which can be either static or dynamic. For instance, if you use a MultiText display element to display the result of an SQL statement, and the query returns several values, all the values will be displayed (depending on the size of the MultiText display element). In contrast, if you use a [Text](#) display element in this case, some text would not be displayed since the Text element can only display one line.

Configure [Multitext properties](#), [Table 27](#), to set behavior and appearance characteristics. The text lines can be aligned horizontally. The LineSpace property which indicates spacing between lines is configurable. An example of a Multitext element is shown in [Figure 73](#).

*Figure 73. Example, Multitext Display Element*

Since the Multitext display element may contain several values (one per line), you can specify a single value for another display element to read by [Indexing MultiText Values](#).

Table 27. Multitext Properties

| Property | Type | Default | Range | Comment |
|------------|---------|-------------|--|---|
| Name | String | MultiText_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not text is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of the text frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of the text frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the text frame. |
| Height | Integer | As created | 5 to 1200 | Height of the text frame. |
| Foreground | Color | 0 | -1 to 127 | Color of the text. |
| Background | Color | 16 | -1 to 127 | Fill color of the frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |

Table 27. Multitext Properties

| Property | Type | Default | Range | Comment |
|---------------------------|---------|---------|--------------------|-----------------------------------|
| DataQuery | Query | | | Script for data retrieval. |
| Horizalign | Enum | Left | Left,Center, Right | Horizontal alignment of the text. |
| Font | Font | 2 | 0 to 71 | Font number. |
| Linespace | Integer | 5 | 0 to 20 | Space between lines |

The following sections provide guidelines for configuring Multitext properties. Refer also to [Basic Properties](#) on page 218.

DataQuery

The DataQuery property can be any one or a combination of the following:

- an SQL statement for a single text string (\$sql...)
- an SQL statement for a multiple text string (\$\$sql....)
- a static user-defined text string

If the DataQuery returns more lines than can be displayed, you may not see all values. To see all values, you can either increase the height of the MultiText display element or limit the number of values returned from the SQL-statement. To limit, either use 'WHERE' clauses, or the maximum number of values option in the sql statement. Refer to [asql](#) on page 291 or [sql](#) on page 317 for details.

When used as a constant text field, line breaks are forced by Carriage Return.

Examples: DataQueries for MultiText. The two last examples provide the same result.

```

$$sql ("SELECT TABLE_NAME FROM ALL_TABLES") ;

"String1";

$sql ("SELECT 'String2' FROM DUAL");

"String" + "3";

"This is
several

```

```
lines
in one
string";

"This is";
"several";
"lines";
"in more";
"strings";
```

Indexing MultiText Values

Since the Multitext display element may contain several values (one per line), you can specify a single value for another display element to read by indexed referencing. The first line is numbered 0, the next 1 and so on.

Example: Read the third line of a MultiText display element

```
$Text_1.DataQuery = $MultiText_1.Value[2];
```

Since the line read from the MultiText can contain space characters, it is recommended that you embed the value in quotation marks (see [Strings](#) on page 281):

Example: Read the third line of a MultiText, and place quotation marks.

```
$Text_1.DataQuery = "|" + $MultiText_1.Value[2] + "|";
```

Referencing an illegal index, such as a negative number or a number greater than the number of lines will result in an empty string.

List

The toolbar button for the List display element is shown in [Figure 74](#).

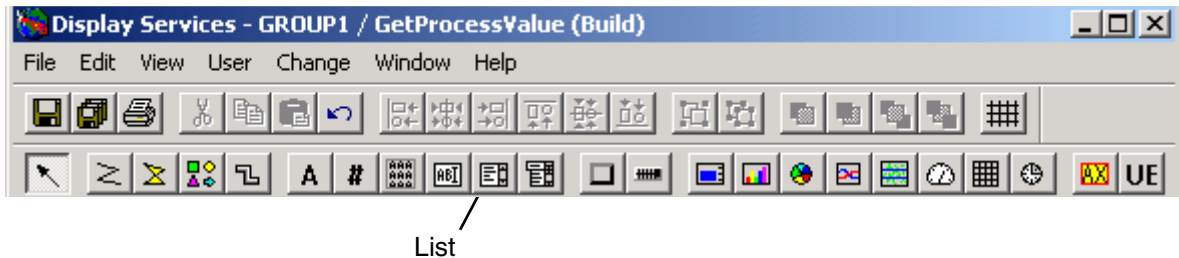


Figure 74. List Button in Toolbar

Configure the [List properties](#), [Table 28](#), to set behavior and appearance properties. The List display element has many of the same properties as the MultiText display element, and it provides several additional features. For instance, you can define actions for the following events:

- selecting or de-selecting an item in the list (clicking once)
- double-clicking on an item in the list

Data for the List display element are retrieved the same way as in the MultiText display element. If the DataQuery returns more values than can be displayed, a scroll bar is added at the right side of the List. To query new list items at runtime, execute a ReQuery for the display element that is querying the list. Specify the ReQuery in the ItemSelected or ItemDoubledClick property of the List display element. An example of this is shown in [ItemSelected & ItemDoubleClicked](#) on page 130.

The ItemCount property indicates the total number of items. An example is shown in [Figure 75](#). If a single line is too wide for the List, a horizontal scrollbar is added at the bottom of the List.

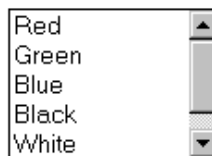


Figure 75. Example, List Display Element

Table 28. List Properties

| Property | Type | Default | Range | Comment |
|------------|---------|-------------|--|---|
| Name | String | MultiText_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the list is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of list frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of list frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the list frame. |
| Height | Integer | As created | 5 to 1200 | Height of the list frame. |
| Foreground | Color | 0 | -1 to 127 | Color of text in the list. |
| Background | Color | 16 | -1 to 127 | Fill color of the list frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |

Table 28. List Properties

| Property | Type | Default | Range | Comment |
|-----------------------------------|---------|---------|-----------------------|--|
| DataQuery | Query | | | Data retrieval script (SQL, dcsub, constant value). |
| SelectedValue | String | | | Currently selected item. |
| SelectedIndex | Integer | -1 | | Position of the SelectedValue in List: -1: None selected 0: First in list 1: Second in list, etc. |
| Font | Font | 2 | 0 to 71 | Font number. |
| ItemSelected | Action | | | Script for action when a list item is selected. |
| ItemDeselected | Action | | | This is not applicable for Windows. |
| ItemDoubleClicked | Action | | | Script for action when list item is clicked twice. |
| ItemCount | Integer | 0 | | Number of items in the list. |
| Sorting | Enum | None | None, Alphabetical | Sorting method of the list entries. |

The following sections provide guidelines for configuring List properties. Refer also to [Basic Properties](#) on page 218.

Height

The height of the list is based on the number of lines. In Run mode, the List calculates the number of visible items as close as possible to the specified height, using fontsize and framewidth. This means that the list may be slightly smaller than you expected, and in some situations it may be slightly taller because of the horizontal scrollbar.

SelectedValue

You can select and deselect one list item at a time. The currently selected item is indicated by the SelectedValue property. If no item is selected, the SelectedValue property is empty.

Example: Reading the selected value from a list

```
$Text_1.DataQuery = $List_1.SelectedValue;
```

For initializing purposes, you can pre-load the SelectedValue property of the List. The value assigned to SelectedValue does not have to be present in the list. When pre-loading the List, the ItemSelected action not is executed and the item is not highlighted.

Example: Pre-loading the List

```
$List_1.SelectedValue = "DEFAULT";  
update("Text_1");
```

ItemSelected & ItemDoubleClicked

These three properties are actions which are executed when you click on an item in the list, or via an execute script, for example: **execute("List_", "ItemSelected")**

- ItemSelected - When you click on an unselected item, the item is highlighted.
- ItemDoubleClicked - When you double-click on an item.

The actions are typically used for updating or notifying other display elements, when the SelectedValue property has changed.

To query new list items at runtime, execute a ReQuery for the display element that is querying the list. This is demonstrated in [Figure 76](#).

First click on the **Events** tab to display the event-type properties, and then select the ItemSelected property. In the Properties Definition window, define the ItemSelected action as shown in [Figure 76](#). This executes a ReQuery for Text_1 and Numeric_1.

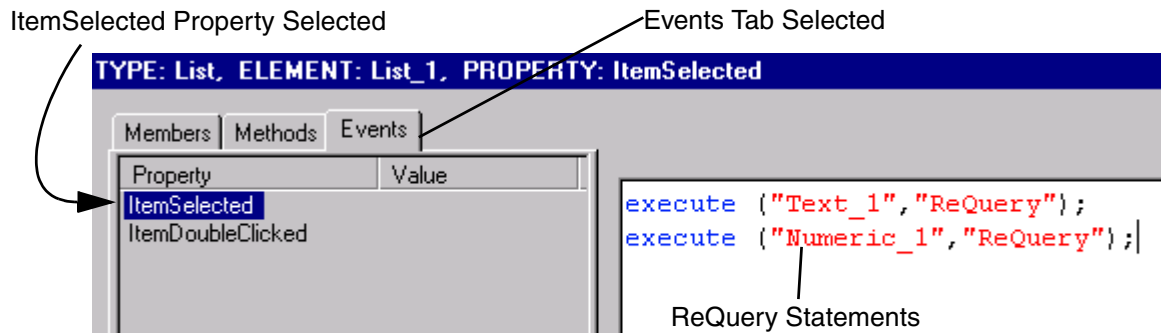


Figure 76. Executing a ReQuery for List Data

A ReQuery is an action that updates a query when the object to be queried has changed. For instance, in this case the data queries for Text_1 and Numeric_1 are executed initially when the display is invoked. The ReQuery causes the text and numeric display elements to get new data whenever a new item is selected from List_1. You MUST execute the ReQuery for both the Text_1 and Numeric_1 in order to change the subscription (get new data). ReQueries are described in greater detail in [ReQuery](#) on page 225.

Matrix

The toolbar button for the Matrix display element is shown in [Figure 77](#).

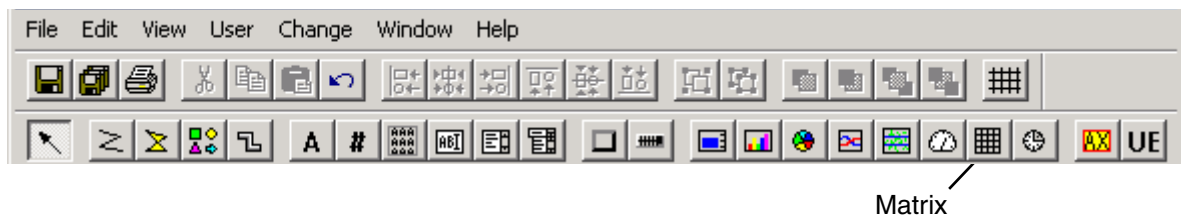


Figure 77. Matrix Button in Toolbar

The data for the Matrix display element is retrieved asynchronously from Oracle or DCS objects of type OpenArray, or from a Trend display element. To query new data at runtime, execute a ReQuery and Update for the Matrix display element. The Matrix display element may also be used to retrieve numeric data from OPC objects

such as softpoints and AC 800M controller objects. The Matrix display element is generally used as a data container and, therefore, is not visible. If set visible it shows the number of rows and columns currently contained in the element.

Some typical applications for the Matrix display element are:

- store data for an [XYPlot](#). To coordinate interaction between the Matrix and XYPlot elements, see [Guidelines for Building an XYPlot](#) on page 133.
- collect data from a [Trend](#) display element for calculations, or to display in tabular format. See [Reading Data from a Trend Display Element](#) on page 137.
- read data from a TCL Unit Array (MOD 300 only). See [Reading Data From a TCL Unit Array](#) on page 135
- apply the **data** function for various read/write transactions as described in:
 - [Modifying or Adding an Entry for a Numeric History Log](#) on page 139
 - [Writing to an Advant Process Object](#) on page 139
 - [Reading From or Writing to an OPC Object](#) on page 140
 - [Querying for the Last History Value](#) on page 141

Configure [Matrix properties](#), [Table 29](#), to set behavior and appearance characteristics.

Table 29. Matrix Properties

| Property | Type | Default | Range | Comment |
|-------------------------|---------|------------|------------|---|
| Name | String | Matrix_# | | Name must be unique within the display. |
| Visible | Enum | True | True/False | Whether or not the matrix is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of matrix frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of matrix frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the matrix frame. |
| Height | Integer | As created | 5 to 1200 | Height of the matrix frame. |

Table 29. Matrix Properties

| Property | Type | Default | Range | Comment |
|-------------|---------|---------|--------------|--|
| Foreground | Color | 0 | -1 to 127 | Color of text in the matrix. |
| Background | Color | 16 | -1 to 127 | Fill color of the matrix frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth |
| DataQuery | Query | | | Script for data retrieval or write(ASQL, dcsub, or Data statement). |
| DataChanged | Action | | | Script for action executed when data from the DataQuery has arrived. |
| Value | | | | Lets you read from or write to a specified cell in the matrix. |
| Column | | | | Lets you read from or write to all rows in a specified column. |
| Rows | Integer | n/a | n/a | Read-only. Number of rows in matrix. |
| Cols | Integer | n/a | n/a | Read-only. Number of columns in matrix. |

The following sections provide guidelines for configuring Matrix properties. Refer also to [Basic Properties](#) on page 218.

Guidelines for Building an XYPlot

The data for each XYPlot curve is retrieved from two matrix elements - one for x-axis data, and one for y-axis data. Therefore, plotting all six curves would require data to be stored in 12 matrix elements.

Use the [DataQuery](#) property to query the data source for the data to be plotted. There are two possible data sources:

- TCL Unit Array (for systems with MOD 300 software) - See [Reading Data From a TCL Unit Array](#) on page 135.

- The Trend Display element - See [Reading Data from a Trend Display Element](#) on page 137.

The XYPlot element retrieves the data from the matrix elements via the [Pn_Xserie](#) and [Pn_Yserie](#) properties in the XYPlot element. How to configure these properties is described in [Data Source](#) on page 214.

Use the [DataChanged](#) property in the Matrix element to update the XYPlot. The XYPlot element requires both a ReQuery action and an Update action. For example:

```
execute ("XYPlot_1", "ReQuery");
update ("XYPlot_1");
```

DataQuery

The DataQuery property can be one of the following:

- an asynchronous SQL statement (asql...), see [asql](#) on page 291.
- an DCS subscription for an OpenArray (dcssub...), see [dcssub](#) on page 300.
- a Data statement. see [data](#) on page 295.

Example:

```
asql ("SELECT NAME, VALUE FROM AI WHERE NAME LIKE FC*")
```

The data returned from this query would be represented in the Matrix element as shown below.

| | |
|--------------|------|
| FC100 | 27.9 |
| FC101 | 32.6 |
| FC102 | 31.1 |
| and so on... | |



When you query an Oracle table, the data type for each matrix column is determined by the data type for the corresponding Oracle table column.

Additional DataQuery examples are provided in:

- [Reading Data From a TCL Unit Array](#) on page 135

- [Reading Data from a Trend Display Element](#) on page 137
- [Modifying or Adding an Entry for a Numeric History Log](#) on page 139
- [Writing to an Advant Process Object](#) on page 139
- [Reading From or Writing to an OPC Object](#) on page 140
- [Querying for the Last History Value](#) on page 141

DataChanged

This action is executed when the data query has retrieved data. Use this action to update any display elements that subscribe to the Matrix data. One application for the Matrix element is to provide data to an XYPlot element. Another application is to retrieve numeric data from an OPC object. Both applications require both a ReQuery action and an Update action. For example:

```
execute ("XYPlot_1", "ReQuery");  
update ("XYPlot_1");
```



The Matrix element is cleared before new data is inserted into it. Each time the Matrix is populated, the number of rows or columns may change. This is another reason to update the applicable display elements when the matrix has new data.

Reading Data From a TCL Unit Array

For systems with MOD 300 software, TCL Unit Array data can be accessed with the Matrix element. The contents of the Unit Array is transferred into a single column of a matrix element. The Unit Array is accessed via the dcsub call with the following syntax:

```
dcsub(<MODUNITNAME.ARRAYNAME>,<TCL Unit Array  
OBJECTYPE>, <ATTRIBUTE>, <RATE>, <LASTELEMENT>)
```

The following are guidelines for accessing data from a unit array:

- Unit Array access always begins at the first element in the array and ends at the element specified by the LASTELEMENT parameter. This does not necessarily have to be the actual last element of the array.
- Even if the Unit array has multiple rows or columns, the data are always presented in a single column in the Matrix element. When you access the data

from the Matrix element you will need to equate the Matrix row number to the corresponding row, column position in the Unit Array. This is illustrated in [Figure 78](#) in [Reading Unit Array Data From the Matrix Display Element](#).

- The Unit Array object types and attributes are listed in [Table 30](#).

Table 30. MOD 300 Unit Array Object Types

| TCL Unit Array Object Types | Attribute |
|-----------------------------|-----------|
| TCL_AI_UNIT_VAR | AIVALUE |
| TCL_AR_UNIT_VAR | ARVALUE |
| TCL_AS_UNIT_VAR | ASVALUE |
| TCL_AB_UNIT_VAR | ABVALUE |
| TCL_AW_UNIT_VAR | AWVALUE |

Example

```
dcssub("UNIT_CA1.ARRSTR_1", "TCL_AR_UNIT_VAR", "ARVALUE",  
dcs_demand, 5)
```

This example will read the Unit array ARRSTR_1 from Unit UNIT_CA1 and get the elements of the array starting at element 0 and ending at element 5.

101.99
105.72
98.16
102.41
100.22
101.10

Reading Unit Array Data From the Matrix Display Element

Data from a Unit Array are stored in a single column in a Matrix element. To equate a Matrix row number with the corresponding Unit Array row/column position, assign each element in the Unit Array a row number in the Matrix starting with one (1) and increasing as you read from the unit array from left to right. This is illustrated in [Figure 78](#).

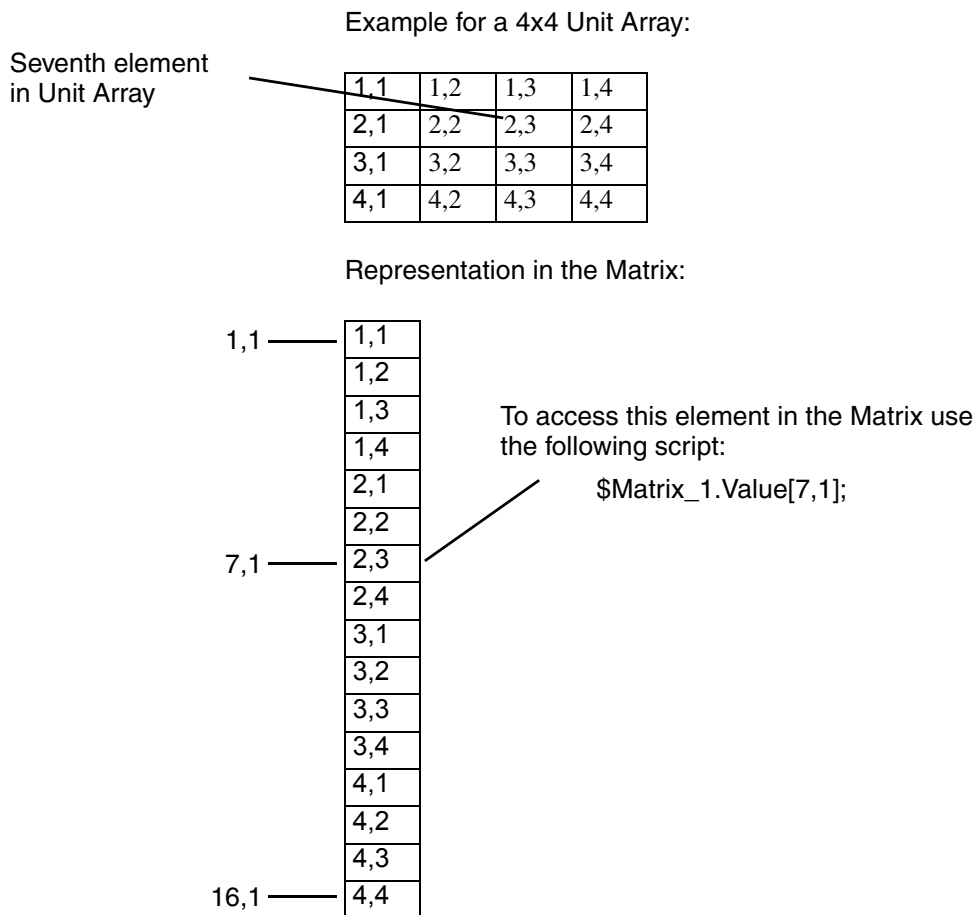


Figure 78. Working with Data from Unit Array

Reading Data from a Trend Display Element

In the DataQuery of the Matrix element enter:

Trend_<xx>[<trend_no>, <time_format>]

where:

Trend_xx is the name of the trend element to get data from.

trend_no is the trend-number (1-6) inside the trend element to read.

time_format is an optional string specifying the format in which the time should be returned. The format follows the syntax of the *time* function described in [time](#) on page 320.

Syntax examples

- To get data from trend #1 in the trend element called T1:
- To get data from trend #3 in the trend element called Trend_1, and also get the time formatted as 'hour:minute:second':

```
T1 [1]
```

```
Trend_1 [3, "%H:%M:%S"]
```

The dataquery is synchronous. When it is finished, DataChanged is executed allowing notification to other elements.

How to Read the Data From the Matrix Display Element

The values are returned in four columns as shown below.

| Column 1 | Column 2 | Column 3 | Column 4 (optional) |
|------------------------|---------------|------------------|-------------------------|
| time seconds (integer) | value (float) | status (integer) | Formatted time (string) |
| . | . | . | . |
| . | . | . | . |

The first column contains the time in seconds.



This value is so large, that if used in float expressions it will cause loss of precision.

The second column contains a value (in float format).

The third column contains the status of the data (OK:0, NoData:101, BadData:102);

The fourth column only contains data if a time format was specified.

The number of rows received can be read by reading '&Matrix.Rows'.

Modifying or Adding an Entry for a Numeric History Log

For these applications, the Data function is used in the DataQuery property to write a new entry value to a specified History log. An example is shown below.

To Modify `data("IMHDA", "ModifyLM", "$HSAITEST,VALUE-1-o", 11.05, 959171516, 0, 0)`

To Add `data("IMHDA", "AddNLog", "$HSAITEST,VALUE-1-o", 11.05, 959171516, 0, 0)`



To add or modify a log entry, the log must reside on the same node where the Display server is running, or be accessible via a remote data provider connected to the Display server

For details regarding the data statement, see the [data](#) on page 295.

Reading Status for ADDNLog and ModifyLM Applications

For single value Add/Modify, if the status is OK, the value is an array with one entry which is the status. If the status is not OK, see [Matrix Structure on an Error](#) on page 143.

For multiple Add/Modify, if the status is OK, the request was carried out. The value returned is an array with a row for each new (or modified) entry. Each row has two columns: one for History status and another for SIP status. Even though the request succeeded, one or more value statuses might indicate an error for that specific log entry. If the status is not OK, the request was not carried out. See [Matrix Structure on an Error](#) on page 143.

Writing to an Advant Process Object

This application is similar to adding/modifying an entry for a History log. The object types that support this functionality are listed in [Table 31](#). For details regarding the Data statement, see [data](#) on page 295.

Example: `data("DCS", "Write", "AITEST", "AI", "VALUE", 105.05)`

Table 31. Supported Objects

| Object Type | Attributes |
|-------------------------------------|--|
| AI | VALUE, HI_LIM1, HI_LIM2, LO_LIM1, LO_LIM2 |
| AO | VALUE |
| DI and DO | STATUS (Write 0/1 - sets bit 8) |
| DAT | R_VAL, IL_VAL, IW_VAL, B0_VAL ⁽¹⁾ |
| TEXT | TEXT |
| PIDCON/PIDCONA ⁽²⁾ | MMI_SP |
| MOD 300 Object Types ⁽³⁾ | - |

- (1) To write Boolean, use B0_VA to write to bit 0; otherwise, use IL_VAL to write all 32 bits.
(2) For systems with Master software only
(3) All MOD 300 Object Types with a single value property are supported.

Reading Status for Process Write Applications

If the status is OK the returned value is a two-dimensional array with one column. This column does not contain any information, but indicates a successful write.

If the status is not OK, see [Matrix Structure on an Error](#) on page 143.

Reading From or Writing to an OPC Object

The syntax for writing to an OPC object is the same as for writing to any other process object, except that object type and attribute are not specified.

Write data("AIPOPC", "Write", "Items:xm5r1ai1", 105.05);

Read data("AIPOPC", "Subscribe", "Items:xm5r1ai1", dcs_3s);

For details on the Data statement, see [data](#) on page 295.

Reading Status for Process Write Applications

If the status is OK the returned value is a two-dimensional array with one column. This column does not contain any information, but indicates a successful write. If the status is not OK, see [Matrix Structure on an Error](#) on page 143.

Reading Process and Status Data for OPC Read Applications

If the status is OK, the returned value is an array with two columns:

- [1,1] holds the value
- [1,2] holds the timestamp

The time stamp is returned as seconds elapsed since January 1, 1970. This value can be converted to a string using the *Time* function.

If the subscription request returns multiple values, the array will contain the corresponding number of rows.

If the status is not OK, see [Matrix Structure on an Error](#) on page 143.

Querying for the Last History Value

If you configure the DATARETRIEVAL user preference for realtime data collection, you can still query for the last History value on a selective basis. This is done via the data statement in the Dataquery property of the Matrix element. There are two forms - one retrieves just the value, the other retrieves the value and time stamp. These two forms are illustrated in the following examples:

Value Only: data("AIPOPC", "SubscribeLHV", "Item1:Value", dcs_3s);

Value & Timestamp: data("AIPOPC", "SubscribeLHVex", "Item1:value", dcs_1s);

For further information regarding the Data statement, see [data](#) on page 295.

Reading Last History Value and Related Status

How the last history data is stored in the matrix depends on whether you queried for value or value and timestamp.

If you queried for value only and the status is OK, then value is stored as an array with one column which holds the value. You can use a script such as the one below in a numeric element to display the last history value stored in the matrix:

```
#Matrix_1.Value[1,1]
```

If you queried for value and timestamp and the status is OK, then value and timestamp are stored as an array of one row five columns containing the following:

```
Value TYPE_FLOAT  
EntryStatus, INTEGER  
ObjectStatus INTEGER  
Timestamp Seconds elapsed since 1/1/1970 INTEGER  
Timestamp Microseconds INTEGER
```

To read the information in this case, use one or more of the following scripts in separate numeric elements:

```
For value: #Matrix_1.Value[1,1]  
For EntryStatus: #Matrix_1.Value[1,2]  
For ObjectStatus: #Matrix_1.Value[1,3]  
For Timestamp in Seconds: #Matrix_1.Value[1,4]  
For Timestamp in Microseconds: #Matrix_1.Value[1,5]
```



If you need to use the timestamp information in another application that requires ASCII data, you can use the *time* function as described in [time](#) on page 320.

If the status is not OK, see [Matrix Structure on an Error](#) on page 143.

For further information regarding last history value, refer to [Subscribing to the Last History Value](#) on page 120

Other Methods for Accessing Data in Matrix Elements

Writing to a Cell

A single cell can be assigned a value with `$Matrix_1.Value[1,2] = "Juice";` and referenced by `$Matrix_1.Value[1,2];`

Access By List, Combobox, and Multitext Elements

List, Combo and MultiText elements can be assigned a Matrix element column with `$$Matrix_1.Column[2]` or the script command `$$columns($Matrix_1, [2, "c10"])` ;

Adding and Deleting Cells

To add or delete cells in a Matrix after the data query, use script commands *insert* (*insert* on page 308) and *delete* (*delete* on page 302).

Matrix Structure on an Error

ADS Errors are generic to the server and typically override the specific error from a data provider. When a data provider request is successfully carried out, the Status parameter of the objects Notify is a single variant value containing the value AOS_OK (0).

When a data provider request fails, an error structure is returned. The status information is a one-dimensional array consisting of three elements. Use the Matrix Value property to access the status as follows:

- `Matrix_n.Value[1,1]` contains the ADS Error value
- `Matrix_n.Value[1,2]` contains the data provider specific error code
- `Matrix_n.Value[1,3]` (if present) contains the error in plain text

ADS-specific error codes are listed in [Table 32](#). General error codes are listed in [Table 33](#).

Table 32. ADS-Specific Error Codes

| Value | Error | Description |
|-------|----------------|---|
| 0 | OK | OK |
| -102 | Dynamic Error | For internal use only. |
| -104 | Syntax Error | Dynamic request matrix does not comply to providers expected format, used in error matrix, typically with an explanation in text in the error matrix. |
| -106 | Provider Error | ADS handled the request, but the data provider did not, e.g. an Oracle or OMF error could be returned. |

Table 32. ADS-Specific Error Codes

| Value | Error | Description |
|-------|-------------------|--|
| -108 | Provider Unknown | Named Data Provider not connected to the Service Provider. |
| -110 | Permission denied | User is not allowed to carry out the request. |

Table 33. General Error Codes

| Value | Error | Description |
|-------|------------------------------------|--|
| 0 | OK | OK. |
| -2 | General error | When no better explanation is available. |
| -3 | General warning | Request carried out, but with a warning. |
| -5 | Request timeout | Timeout occurred before request has answered. |
| -7 | Request is pending | Further status can be expected. |
| -12 | Unknown object | The requested object name can not be found. |
| -14 | Unknown object type | No such object type is known. |
| -16 | Unknown object attribute | Specified object does not support the specified attribute. |
| -18 | Illegal event subscription | An event subscription can not be made. |
| -22 | Unknown value type | Type (integer, float, and string). |
| -24 | Illegal value size | For example, too long a string. |
| -26 | Illegal value type | Value type known, but not allowed where used. |
| -28 | Out of memory | |
| -32 | Illegal cyclic request time | The specified interval is illegal or unsupported. |
| -42 | Illegal object information request | The request made is not supported. |
| -52 | Unknown log request | For example, OMF operation not supported) |

Table 33. General Error Codes

| Value | Error | Description |
|-------|----------------------------|--|
| -54 | Unknown log request method | When accessing a log through ADSdpLOG, a calculation method is specified (mean, max, min, etc.) If the calculation method is unknown, this error is returned. |
| -56 | Unknown log collection | When accessing a log through ADSdpLOG, a data collection method is specified ('D':(hsDISPLAY+hsINST), 'R' or 'I':(hsRAW+hsINST)) If the data collection method is unknown, this error is returned. |
| -58 | Unknown log time selection | When accessing a log through ADSdpLOG, a time selection is specified ('U':(hsEND_TIME+hsUTC), 'T':(hsEND_TIME+hsPRESENT), 'F':(hsSTART_TIME+hsUTC), 'S':(hsFROM_TIME+hsUTC)) If the time selection is unknown, this error is returned. |
| -60 | Object is inactive | |
| -62 | Request is unsupported | Data Provider does not support this request. |
| -63 | Returned data is truncated | Request resulted in more data than could be returned. |
| -110 | Permission denied | Request denied by Data Provider or user rights (preference setting) |
| -202 | Bad timer request | See dpTIM for correct format. |

Edit

The toolbar button for the Edit display element is shown in [Figure 79](#).

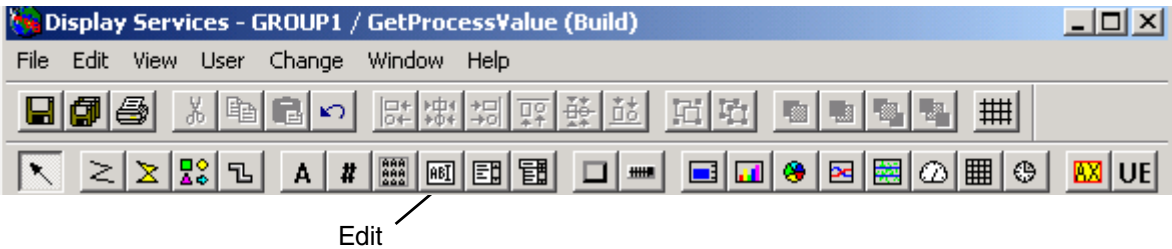


Figure 79. Edit Button in Toolbar

The Edit display element is an interactive text field where you can enter text data. You can then specify an action to execute when the entry is made. An example of the Edit display element is shown in Figure 80.



Figure 80. Example, Edit Display Element

Configure [Edit properties](#), [Table 34](#), to set behavior and appearance characteristics. You can use the Edit display element in either single-line or multi-line mode. In single-line mode, you can enter just one line of data. If the line exceeds the width of the Edit field, the text is automatically scrolled. In multi-line mode several lines can be entered, causing the Edit field to scroll both horizontally and vertically, if the size is exceeded.

To edit an Edit display element, click on it, navigate within the field using the cursor keys, and then highlight text using SHIFT + the cursor keys. You can define an action to be executed after the ENTER key is pressed while the Edit display element has input focus.

Table 34. Edit Properties

| Property | Type | Default | Range | Comment |
|----------|--------|---------|-----------|---|
| Name | String | Edit_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |

Table 34. Edit Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--|---|
| Visible | Enum | True | True/False | Whether or not the field is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of edit frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of edit frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the edit frame. |
| Height | Integer | As created | 5 to 1200 | Height of the edit frame. |
| Foreground | Color | 0 | -1 to 127 | Color of the text. |
| Background | Color | 16 | -1 to 127 | Fill color of the edit frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| DataQuery | Query | | | Script for data retrieval (SQL, dcsub, or constant value). |
| Value | String | | | Result of the DataQuery. |
| GotFocus | Action | | | Script for action to execute when edit field gets input focus. |

Table 34. Edit Properties

| Property | Type | Default | Range | Comment |
|-----------|---------|------------|--------------------------------|---|
| LostFocus | Action | | | Script for action to execute when edit field loses input focus. |
| Activate | Action | | | Script for action to execute when the ENTER key is pressed in the edit field. This is only applicable In single line edit mode. |
| Status | Enum | OutOfFocus | OutOfFocus, InFocus, Activated | The latest focus state of the edit field. The Activated state is only applicable for single line edit mode. |
| EditType | Enum | Single | Single, Multi | Single- or multi-line edit mode. |
| Font | Font | 2 | 0 to 71 | Font number. |
| MaxLength | Integer | 0 | | Max number of characters the user is allowed to enter in Run mode, zero = disabled. |
| Nexttab | String | | | Name of the next Edit element to get focus when the TAB key is pressed. PC-Client only. |

The following sections provide guidelines for configuring Edit properties. Refer also to [Basic Properties](#) on page 218.

DataQuery

The DataQuery result is stored in the Value property, and is displayed in the Edit field. This is the property that other display elements can reference to read the data entered in the Edit field.

Activate

The Activate property is an action which is typically used to update other display elements with the data stored in the Value property. This action is only applicable for the single-line mode. The action is executed when the ENTER key is pressed.

Example: This Activate action transfers the data entered in the edit field to a Text display element, after the ENTER key is pressed.

```
$Text_1.DataQuery = "|" + $Edit_1.Value + "|";  
update ("Text_1");
```

GotFocus & LostFocus

The GotFocus action is executed when you select (click on) the Edit field. The LostFocus action is executed when you select another display element, or the entire window loses focus.

Status

The Status property indicates the last state of the Edit field. Possible states are:

- InFocus: Edit field is currently in focus.
- OutOfFocus: Edit field is currently not in focus.
- Activated: ENTER key pressed with Edit field in focus. Only when EditType = Single.

MaxLength

This property controls how many characters you are allowed to enter in the Edit field. The value 0 means that the length check is disabled. If the Edit field is in multi-line mode, each new line counts as a character. If for instance, if MaxLength = 4, you would be allowed to enter:

- Single mode: ABCD
- Multi mode (new line counts as a character):

```
AB  
C
```

Combobox

The toolbar button for the Combobox display element is shown in [Figure 81](#).

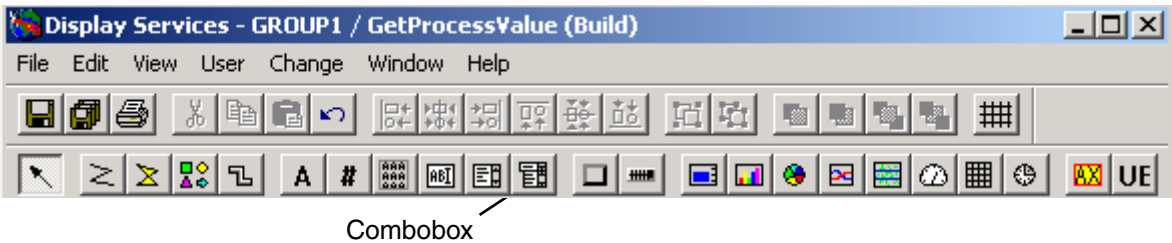


Figure 81. Combobox Button in Toolbar

The Combobox combines the functions of an edit field, a list, and a button. An example is shown in Figure 82. Data can be entered directly into the edit field, or you can select from a list of items which is displayed by clicking the arrow-button. Further, you can specify actions to be executed when an item is selected, or when the display element has keyboard focus, or loses keyboard focus.

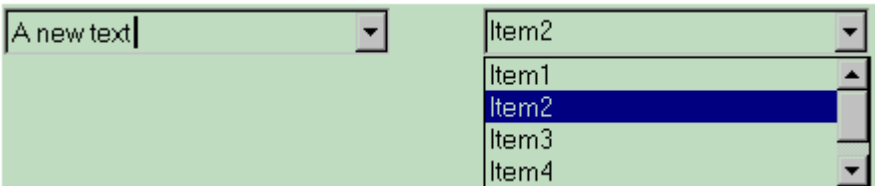


Figure 82. Example, Combobox in Selection Mode

Configure Combobox properties, Table 35, to set behavior and appearance characteristics.

Table 35. Combobox Properties

| Property | Type | Default | Range | Comment |
|----------|--------|------------|------------|--|
| Name | String | Combobox_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the field is visible in Run mode. |

Table 35. Combobox Properties

| Property | Type | Default | Range | Comment |
|---------------|---------|------------|--|--|
| X | Integer | As created | 0 to 1200 | X coordinate of combobox frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of combobox frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the combobox frame |
| Height | Integer | As created | 5 to 1200 | Height of the Combobox frame. |
| Foreground | Color | 0 | -1 to 127 | Color of text in Combobox. |
| Background | Color | 16 | -1 to 127 | Fill color of the Combobox frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| DataQuery | Query | | | Script for data retrieval. |
| SelectedValue | String | | | Currently selected value. |
| SelectedIndex | Integer | -1 | | Currently selected value. -1: Initially or when user enters data 0: First selected 1: Second selected, etc. |
| Font | Font | 1 | 0 to 71 | Font number. |

Table 35. Combobox Properties

| Property | Type | Default | Range | Comment |
|------------------|---------|---------|-------------|---|
| VisibleItemCount | Integer | 5 | | Number of items in the list. |
| Editable | Enum | True | True, False | Determines whether user can edit the Combobox. |
| ItemSelected | Action | | | Script for action when list item is selected. |
| GotFocus | Action | | | Script for action when Combobox gets input focus. |
| LostFocus | Action | | | Script for action when Combobox loses input focus. |
| DropDown | Action | | | Script for action to execute when the arrow down button of Combobox is activated. |

The following sections provide guidelines for configuring Combobox properties. Refer also to [Basic Properties](#) on page 218.

VisibleItemCount

This indicates the height of the list. If there are more items than can be shown, a scrollbar is provided. List height is calculated from the VisibleItemCount, so height will vary with fontsize.

Editable

The Editable property indicates whether or not you have permission to edit the value in the edit field. If you try to select the value in the edit field with Editable set to false, the value will be highlighted but data can not be entered or deleted.

ItemSelected

This property is an action which is executed when you either select an item from the selection list or press the ENTER key in the edit field.

Button

The toolbar button for the Button display element is shown in [Figure 83](#).

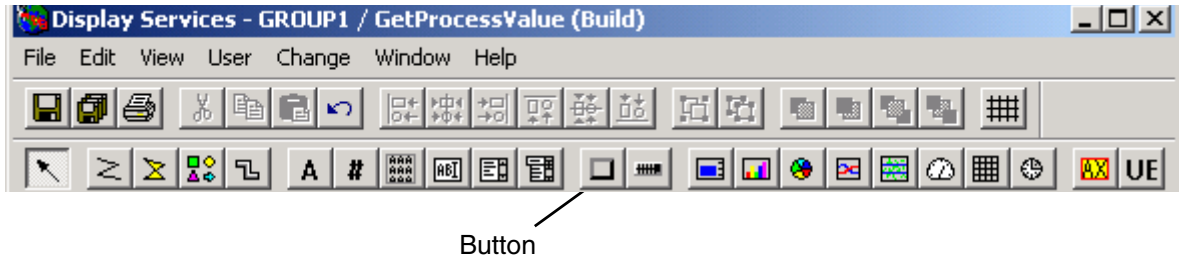


Figure 83. Button Button in Toolbar

The button display element is used to execute user-defined actions such as:

- Execute a system command
- Switch from one display to another
- Print a display

Some examples of buttons are shown in [Figure 84](#).

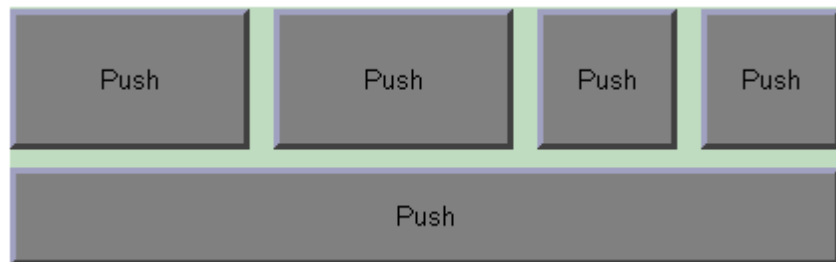


Figure 84. Examples of Buttons

Configure [Button properties](#), [Table 36](#), to set behavior and appearance characteristics.

Table 36. Button Properties

| Property | Type | Default | Range | Comment |
|------------|----------|------------|--------------|---|
| Name | String | Button_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the button is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of the button frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of the button frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the button frame. |
| Height | Integer | As created | 5 to 1200 | Height of the button frame. |
| Foreground | Color | 0 | -1 to 127 | Color of caption. |
| Background | Color | 16 | -1 to 127 | Fill color of the button. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Type | Enum | Push | Push/Toggle | Push: On when activated, else Off. Toggle: Keeps On/Off state until next time activated. |
| Action | Action | | | Script for action to execute when activated. |
| Caption | Textline | Button | | Button label in run mode. |
| Font | Font | 0 | 0 to 71 | Font of the caption. |
| Status | Integer | 0 | 0,1 | Button Status. In toggle mode, 1 = activated, 0 = deactivated |

The following sections provide guidelines for configuring Button properties. Refer also to [Basic Properties](#) on page 218.

Type

The Type property establishes the type of button:

| | |
|--------|--|
| Push | Executes action when activated by the pointing device. The button stays activated as long as it is pushed, and returns to the initial state when released. |
| Toggle | Remains in the state it is toggled to, until it is toggled back. |

Status

The status property indicates the active state of the Toggle button. This property is not applicable for Push buttons. If Status = 0 the Button is deactivated (out), and if the status = 1 the Button is activated (in). The Status property can either be set or read. You can use the Status property to activate and deactivate the button from another display element.

Visible

If the Visible property of a Button is false, the Action will not execute when you click on the button. However, the Action can be executed by a command from another display element.

To configure an invisible button that responds when you click on it, set the Foreground and Background properties to Transparent, and the FrameWidth to zero. See [Visible](#) on page 220.

ABBBUTTON

The toolbar button for the ABBButton display element is shown in [Figure 85](#).

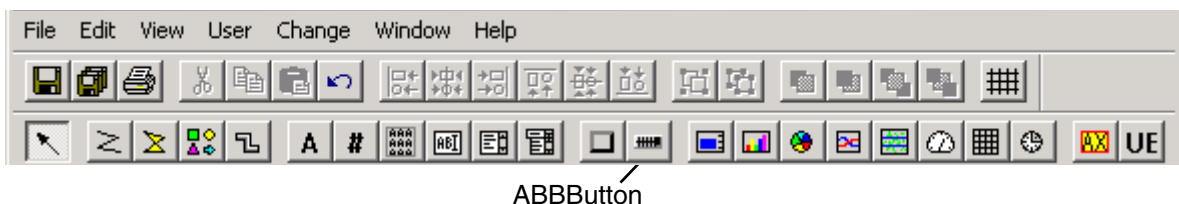


Figure 85. ABBButton Button in Toolbar

The ABBButton display element is a collection of ten buttons and the ABB logo. Each of the ten buttons has the same properties as the Button display element described above, but one specified font size applies to all ten buttons. An example is shown in [Figure 86](#).



Figure 86. Example, ABBButton

Configure the [ABBBUTTON properties](#), [Table 37](#) to set behavior and appearance characteristics.

Table 37. ABBButton Properties

| Property | Type | Default | Range | Comment |
|----------------------------|-----------------------|-------------|-------------|--|
| Name | String | ABBBUTTON_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the ABBButton is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of ABBButton frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of ABBButton frame's upper left corner. |
| Width | Integer | As created | 200 to 1200 | Width of the ABBButton frame. |
| Height | Integer | As created | 100 to 1200 | Height of the ABBButton frame. |
| Foreground | Color | 8 | -1 to 127 | Color of the captions. |
| Framewidth | Integer | 3 | | Width of the Motif frame. |

Table 37. ABBButton Properties

| Property | Type | Default | Range | Comment |
|------------|----------|---------|---------------|---|
| Font | Font | 3 | 0 to 71 | Font used for captions. |
| Fn_Caption | Textline | | | Caption for button number n . $n = (1 \text{ to } 10)$ If no caption is defined for a button, the button will be dimmed to indicate it is not functional |
| Fn_Action | Action | | | Script for action of button number n when activated. |
| Fn_Type | Enum | Push | Push / Toggle | Button type for button number n . Push: On when activated, else Off. Toggle: Keeps On/Off state until next time toggled. |
| Fn_Status | Integer | 0 | 0,1 | Status of button number n . In toggle mode, 1 = activated, 0 = deactivated |

Foreground

The only configurable color property of the ABBButton is the foreground color. This applies to the user-defined Captions only. The F1 to F10 text and the ABB logo are not affected.

Bar

The toolbar button for the Bar display element is shown in [Figure 87](#).

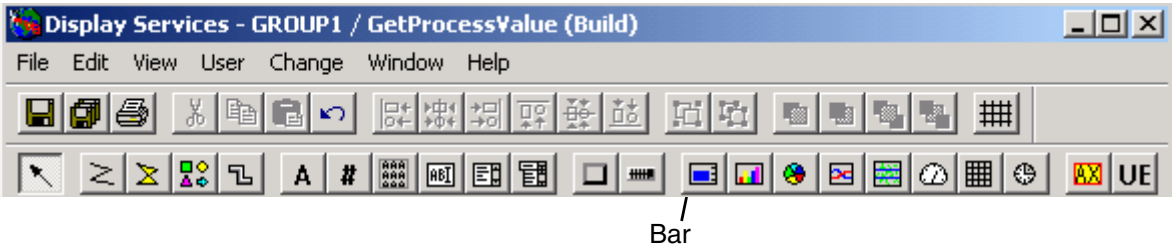


Figure 87. Bar Button in Toolbar

The Bar display element provides a vertical bar graph. It may also show a scale, a numeric field and a trend curve. An example Bar display element is shown in Figure 88.

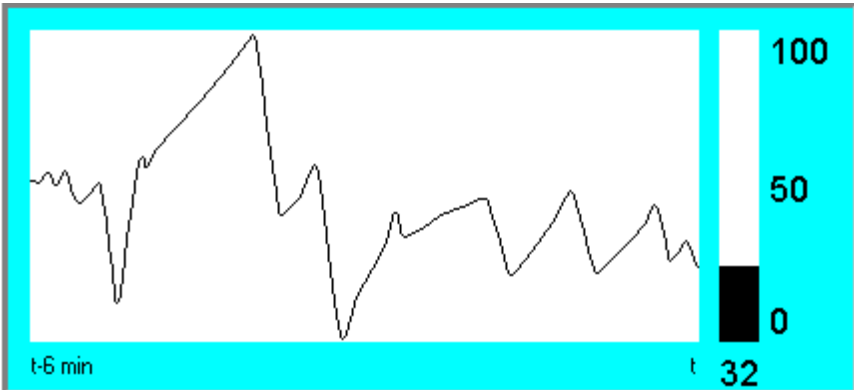


Figure 88. Example Bar Display Element

Configure the Bar properties, Table 38, to set behavior and appearance characteristics.

Table 38. Bar Properties

| Property | Type | Default | Range | Comment |
|----------|--------|---------|-----------|---|
| Name | String | Bar_1 | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |

Table 38. Bar Properties

| Property | Type | Default | Range | Comment |
|-------------|---------|------------|--------------|--|
| Visible | Enum | True | True/False | Whether or not bar is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of bar frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of bar frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the bar frame. |
| Height | Integer | As created | 5 to 1200 | Height of the bar frame. |
| Foreground | Color | 66 | -1 to 127 | Color of border, scale, trend, and bar. |
| Background | Color | 16 | -1 to 127 | Fill color of the bar frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |
| DataChanged | Action | | | Script for action to execute when Value changes. |
| Action | Action | | | Script for action to execute when you click anywhere inside the bar. |
| Value | Float | | | The result of the DataQuery |

Table 38. Bar Properties

| Property | Type | Default | Range | Comment |
|---------------------------|----------|---------|--|---|
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| Scalemax | Float | 100 | >ScaleMin | Greater than ScaleMin |
| Scalemin | Float | 0 | <ScaleMax | Less than ScaleMax |
| ScaleFormat | Textline | %.0f | see Format on page 118 | Format of numeric presentation. |
| ScaleFont | Font | 3 | 0 to 71 | Fontsize for scale. |
| Scale | Enum | True | True, False | When true, scale is shown, when false scale is hidden. |
| TrendBack-Ground | Color | 0 | 0 to 127 | Color of the trend background. |
| Trend | Enum | True | True,False | When true, trend is shown, when false trend is hidden. |
| NumericFont | Font | 3 | 0 to 71 | Font size for numeric field. |
| Numeric | Enum | True | True,False | When true, numeric field is shown, when false numeric field is hidden. |
| DataQuery | Query | | | Script for data retrieval (SQL, dcsub, or constant value). |

The following sections provide guidelines for configuring Bar properties. Refer also to [Basic Properties](#) on page 218.

Scale

The scale is displayed to the right of the vertical bar. It shows the user-defined minimum (Scalemin property) and maximum (Scalemax property) scale values respectively at the bottom and the top.

The font for scale values is set by the ScaleFont property, and the format for scale values is set by the ScaleFormat property (same format as Numeric as described in [Numeric](#) on page 115).

Whether or not the scale is displayed is determined by setting the Scale property either True (visible) or False (invisible).

Numeric

The numeric field indicates the current value of the bar, and is displayed below the bar. You can specify the font for the Numeric field by setting the NumericFont property. The format follows the ScaleFont format.

Whether or not the numeric field is displayed is determined by setting the Numeric property either True (visible) or False (invisible).

Trend

The trend curve shows the most current 100 values from the bar graph, and is displayed to the left of the bar. A scale is provided at the bottom of the trend curve, starting at current time (right) and ending (left) at current time minus (100 * update frequency).

The trend curve is drawn in the foreground color. You can select the background color of the trend area by setting the TrendBackGround property.

The values shown in the trend curve are temporary values, meaning that each time you switch to the display, the trend curve area will be empty.

Whether or not the trend curve is displayed is determined by setting the Trend property either True (visible) or False (invisible).

Multibar

The toolbar button for the Multibar display element is shown in [Figure 89](#).

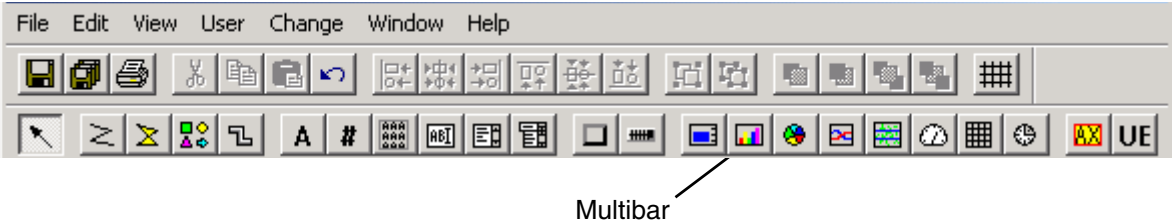


Figure 89. Multibar Button

Each Multibar display element, [Figure 90](#), may have up to 35 individually colored bar graphs.

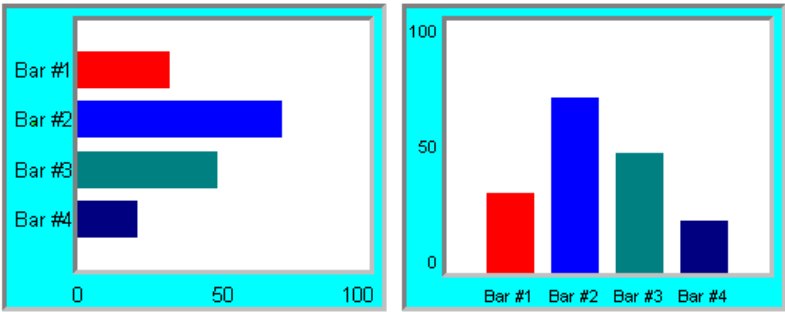


Figure 90. Example, Two Multibar Display Elements

Configure the [Multibar properties](#), [Table 39](#), to set behavior and appearance characteristics. The direction of the multibar can be either vertical or horizontal. You can show grid lines either vertically or horizontally, and with user-defined spacing and colors.

To create individual bars in the Multibar display element, enter a caption and a DataQuery for each bar. Invalid DataQueries are indicated by a cross on the applicable bar. Within a Multibar display element, all bars must be positioned in the same direction.

Vertical bars do not require captions. Captions must be provided for horizontal bars; however, they are not required to be visible. The MultiBar has a minimum size which is calculated from the fontsize and number of bars.

Table 39. MultiBar Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--------------|---|
| Name | String | MultiBar_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the bar is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of multibar frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of multibar frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the multibar frame. |
| Height | Integer | As created | 5 to 1200 | Height of the multibar frame. |
| Foreground | Color | 8 | -1 to 127 | Color of border, scale. |
| Background | Color | 16 | 0 to 127 | Fill color of the multibar frame. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |

Table 39. MultiBar Properties

| Property | Type | Default | Range | Comment |
|----------------|---------|-------------|--|--|
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| Direction | Enum | Horizontal | Horizontal, Vertical | Direction of the bars. |
| Scalemax | Float | 100.0 | >ScaleMin | Max scale of all bars. |
| Scalemin | Float | 0.0 | <ScaleMax | Min scale of all bars. |
| BarType | Enum | SingleColor | SingleColor, MultiColor | Bar coloring method: SingleColor - All bars are using BarColor. MultiColor - Each bar is individually colored, using Bn_Color. |
| BarBackground | Color | 0 | -1 to 127 | Color of the Bar Background. |
| BarColor | Color | 20 | -1 to 127 | Color of the bars. |
| BarFrame-width | Integer | 3 | 0 to 10 | Width of the 'inner' frame. |
| BarFramestate | Enum | In | In,Out | The visual state of the internal Motif-frame. |
| ShowScale | Enum | True | True, False | Controls if the scale should be visible or not. |

Table 39. MultiBar Properties

| Property | Type | Default | Range | Comment |
|-----------------------------|----------|---------|--|--|
| ShowCaption | Enum | True | True, False | Controls if the captions for each bar should be displayed. |
| Grid | Integer | 0 | | Number of grid lines. Minimum is 20. |
| GridColor | Color | 20 | -1 to 127 | Color of the Grid lines. |
| ScaleFormat | Textline | %.0f | See Format on page 118 | Format of numeric presentation. |
| SortOrder | Enum | NoSort | NoSort, LowFirst, HighFirst | Ordering of bars in window. See SortOrder on page 166. |
| SelectedBar | Integer | 0 | 0 to 35 | The number of the last selected bar, 0 = none selected. |
| Action | Action | | | Script for action when you click inside the MultiBar. |
| Font | Font | 3 | 0 to 71 | Font used in scale and labels. |
| Bn_Caption | Textline | | | Label for Bar number n ($n=1$ to 35) |
| Bn_Data_Query | Query | | | Script for data retrieval for bar number n ($n=1$ to 35). Data Type is Float. |
| Bn_Value | Float | | | The result of the Bn_DataQuery. |
| Bn_Color | Color | 20 | -1 to 127 | Color of the Bar number n , when BarType is MultiColor. |

The following sections provide guidelines for configuring Multibar properties. Refer also to [Basic Properties](#) on page 218.

BarType

BarType determines whether all the bars should have the same color or use individual colors:

- SingleColor When SingleColor is selected, all bars are displayed using the color specified by common BarColor property, regardless of individual Bn_Color settings.
- MultiColor When MultiColor is selected, all bars are displayed using the color specified by the individual Bn_Color properties, rather than the common BarColor property.

SortOrder

If more than one bar graph is displayed, you can select the order of appearance:

- NoSort Bar graphs are displayed in the same order as they were entered.
- HighFirst If horizontal, bar graphs are displayed in decreasing order from top to bottom. If vertical, bar graphs are displayed in decreasing order from left to right.
- LowFirst If horizontal, bar graphs are displayed in increasing order from top to bottom. If vertical, bar graphs are displayed in increasing order from left to right.

SelectedBar & Action

The Action and SelectedBar properties are closely related. When a bar in the Multibar display element is clicked, the SelectedBar property is set to the bar number that was clicked and the action specified in the Action property is executed. If any part of the element other than a bar is clicked, the SelectedBar property is set to 0, and the Action is executed. [Figure 91](#) and [Figure 92](#) show the active bar areas. The active area extends to left and right of the bar, including caption as indicated by shading. White areas are not considered part of a bar.

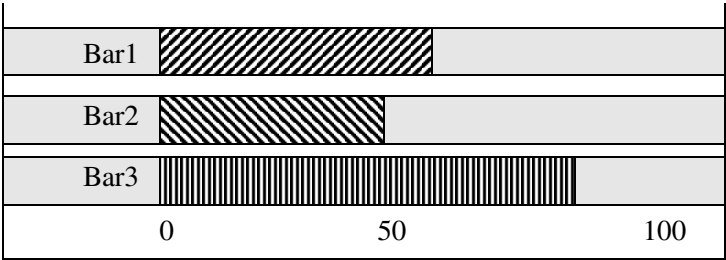


Figure 91. Active bar areas when Direction = Horizontal

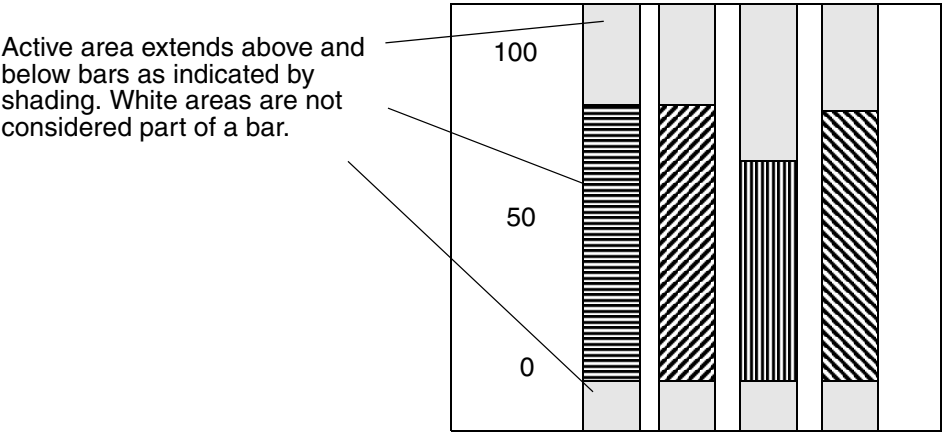
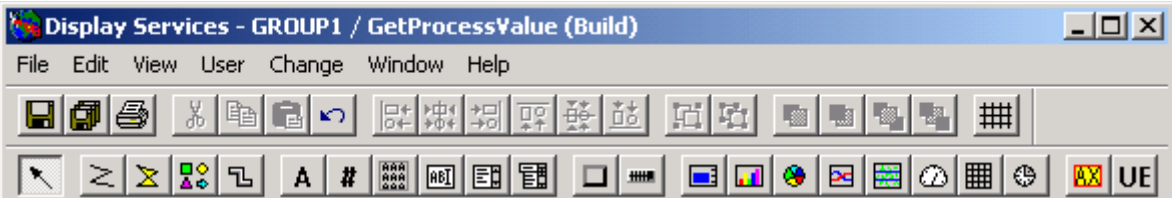


Figure 92. Active bar areas when Direction = Vertical

Pie

The toolbar button for the Pie display element is shown in [Figure 93](#).



Pie

Figure 93. Pie Button

The pie display element represents a pie chart with up to ten slices. The whole pie (100%) is the sum of all inputs. Configure the [Pie properties](#), [Table 40](#), to set behavior and appearance characteristics.

The pie chart is dynamically updated. The period is specified in the Update field. Each slice requires a Caption and a DataQuery. If you define a query that is not valid, a cross is displayed on the applicable slice.

An example of a pie chart is shown in [Figure 94](#).



Figure 94. Example, Pie Chart

Table 40. Pie Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--|---|
| Name | String | Pie_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not the pie is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of pie frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of pie frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the pie frame. |
| Height | Integer | As created | 5 to 1200 | Height of the pie frame. |
| Foreground | Color | 0 | -1 to 127 | Color of border. |
| Background | Color | 16 | -1 to 127 | Fill color of the pie chart. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |

Table 40. Pie Properties

| Property | Type | Default | Range | Comment |
|---------------------------|----------|---------|-----------------------|---|
| Location | Enum | Left | Left,Top,Right,Bottom | Location of the pie in relation to the caption. |
| Font | Font | 1 | 0 to 71 | Font for captions. |
| P _n _Caption | Textline | | | Name of pie slice number <i>n</i> . (<i>n</i> =1 to 10) |
| P _n _Dataquery | Query | | | Script for data retrieval. |
| P _n _Color | Color | n + 63 | -1 to 127 | Color of pieslice number <i>n</i> . (<i>n</i> =1 to 10). |

Location

You can specify a text string and color for each slice, and specify where the pie will be displayed, with respect to the text. The choices are up, down, left and right. For example, selecting Up places the pie in the top part of the display element, and the text underneath the pie.

Trend

The toolbar button for the Trend display element is shown in [Figure 95](#).

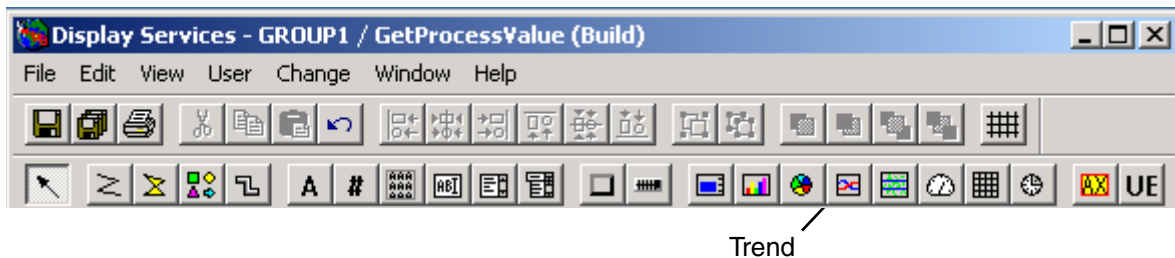


Figure 95. Trend Button

The Trend display element displays historical data for logs configured in History Services and/or TTD logs in systems with Master software, [Figure 96](#). Each trend

element can have up to six trends. This element has features for viewing the trends such as filtration, zooming, placing a ruler, and changing the scope in run-time.

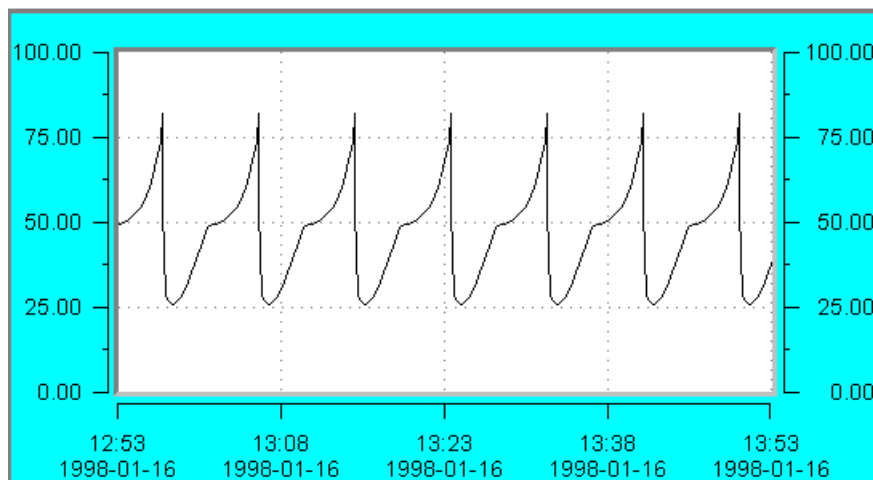


Figure 96. Example, Trend Display Element

Configure the [Trend properties](#), [Table 37](#), to set behavior and appearance characteristics.

Trend Data

Data can be retrieved from a combination of the following data source types:

- OPCHDA - OPC server
- LOGMGR - History logs/TTD on earlier Enterprise Historian platforms
- Oracle - SQL query
- ALM object - in special applications using Master software

The trend element can display up to 500 points, regardless of the data source. For LOGMGR and ALM data sources, if there are more than 500 or less than 500 points, the log manager interpolates or extrapolates as required to make the trend graph. For SQL data sources, the SQL data provider returns just the first 500 values.

For ALM and LOGMGR data sources, if you specify the composite log by access name rather than the full log name, the query will default to a log that uses either an

instantaneous or average calculation. To specify a log that performs a different calculation without having to know the full log name, specify the calculation using the [treatment](#) property.

Trend Properties Overview

The trend element can function independently; however, to use the advanced functions mentioned above, it must interact with other display elements and some scripting will be needed. Refer to [Section 6, Display Scripting](#).

The Trend display element type has a number of properties common to all trend curves, in order to set such characteristics as:

- font and color of the timescale
- visibility and color of vertical- and horizontal dotted grid lines
- rulercolor

These properties apply to individual trend curves:

- font, color, scale min/max, linewidth and visibility
- representation of each trend (linear, sample/hold, poles)
- filterfactor, used for the filtrating algorithm
- timeoffset, used to delay a single trend

Some properties are read-only, and are read by other display elements, to display such data as:

- current value for a trend
- ruler time and value for a trend

Finally some properties are used for invoking actions such as changing scope, zooming, and so on, and for supplying values used for the actions. These properties include:

- percentage to zoom in or out
- number of steps to move the ruler
- which trend is currently selected

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|------------|---------|------------|--------------|--|
| Name | String | Type_# | | Name must be unique in the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True,False | Whether or not the trend display element is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of the trend frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of the trend frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the trend frame. |
| Height | Integer | As created | 5 to 1200 | Height of the trend frame. |
| Foreground | Color | 0 | -1 to 127 | Color of border. |
| Background | Color | 16 | -1 to 127 | Background color of the trend graph. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | Used to show depth. |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|----------------|---------|--|--|---|
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| ScopeEndTime | String | Current time: %H:%M | | End Time for the visible scope. |
| ScopeStartTime | String | Current time minus 1 hour: %H:%M | | Start Time for the visible scope. |
| ScopeEndDate | String | Current date: %Y-%m-%d | | End Date for the visible scope. |
| ScopeStartDate | String | Current date: %Y-%m-%d | | Start Date for the visible scope. |
| Scope | Numeric | 3600 (one hour) | | Duration of the trend. Format is hh:mm:ss |
| ScopeLockColor | Color | 66 | -1 to 127 | The color of the frame surrounding the trend area when zooming. |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|-----------------|---------|-------------------|-----------|--|
| ScopeAdjustment | Numeric | 0 | | Factor used to shift the scope forward or backward with the <i>ChangeScope</i> TrendMode-command. The value of the scope property is multiplied by the specified number, and this result is added to the stop time. A negative number will move back in time, and positive will move forward. |
| TrendBackground | Color | 0 | -1 to 127 | Color of the curve area. Default is black. |
| RulerColor | Color | 31 | -1 to 127 | Color of the ruler. Default is white. |
| RulerBehaviour | Enum | AID | AID, OS | The behavior of the ruler, see Ruler & RulerBehaviour on page 185. |
| RulerTimeFormat | String | %m-%d %H:%M:%S | | The format for the RulerValue property, when the ruler is used. |
| RulerAction | String | | | Script for action executed each time the ruler is placed in the curve area. Typically used to update other display elements. |
| RulerStep | Numeric | 0 | | Number of resolution steps to move the ruler. Negative numbers move left, positive moves right. |
| ZoomPercent | Numeric | 0 | | Size of the scope in percent after zooming, when using TrendMode 'ZoomInOut'. For example, with ZoomPercent = 80, a scope of 1 hour will be 48 min. after zooming. |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|----------------------|--------|----------|--|--|
| VerticalGrid | Enum | True | True,False | If True, the vertical grid lines are visible. |
| HorizontalGrid | Enum | True | True,False | If True, the horizontal grid lines are visible. |
| GridColor | Color | 14 | -1 to 127 | Color of the grid lines. |
| DataSource | Enum | LOGMGR | LOGMGR, SQL, ALM, OPCHDA | Method for retrieving trend data. |
| ShowTime | Enum | True | True,False | If True, the time scale and time ticks are visible. If False, the curve areas are recalculated to include the area of the time scale. |
| ShowLabels | Enum | True | True,False | If True, all Y-scale labels are visible, provided the corresponding Tn_Visible is True. If False, all labels are invisible, and the curve areas are recalculated. |
| TrendMode | Enum | Normal | See TrendMode on page 186. | Sets the mode of the display element from other display elements. Used when zooming, changing scope and querying new data. |
| TimeScaleColor | Color | 0 | -1 to 127 | Color of the time labels. |
| TimeScaleFont | Font | 1 | 1 to 71 | Font for time labels. |
| TimeScaleTime Format | String | %H:%M | | The time format of the X-axis. |
| TimeScaleDate Format | Font | %Y-%m-%d | | The date format of the X-axis |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|---------------|---------|---------|--|---|
| SelectedTrend | Numeric | 1 | 1 to 6 | Number of currently selected trend. |
| NoDataColor | Color | 69 | -1 to 127 | Color of trend curve for no data. |
| BadDataColor | Color | 65 | -1 to 127 | Color of trend curve for bad data. |
| GapToEdge | Integer | 0 | 0 - 10 | Gap from current time to the left border of the trend area, specified in percent of the width of the trend area. |
| Tn_Caption | String | | | User defined label. Must be supplied, but does not have to be visible. |
| Tn_Font | Font | 1 | 1 to 71 | Font for the Y-label for each trend. |
| Tn_Channel | Numeric | | | Data provider channel number. |
| Tn_ObjectName | String | | | When DataSource is LOGMGR, OPCHDA, or ALM, this is the name of the log object (access name of log as assigned in History). |
| Tn_LogName | String | | | When DataSource is OPCHDA, enter the data provider name (-name argument), for example AIPHDA or IMHDA. When DataSource is SQL, use this to enter the SQL query. |
| Tn_DataSource | String | COMMON | ALM, SQL, LOGMGR, OPCHDA, COMMON | This lets you specify a different type of data source for each trace trend number n (1 to 6). If set to COMMON, the specification in the DataSource property is used. |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|-------------------|---------|--------------|---|--|
| Tn_Attribute | String | | | Logged attribute of the object (only applicable when DataSource is ALM). |
| Tn_Description | String | | | Description of the object. |
| Tn_Unit | String | | | Engineering units of the trended data. |
| Tn_Scalemax | Numeric | 100 | | Maximum shown value for a trend. |
| Tn_Scalemin | Numeric | 0 | | Minimum shown value for a trend. |
| Tn_ValueFormat | String | %.2f | | Format for the Y-labels of a trend. |
| Tn_Color | Color | 1 | -1 to 127 | Color for each trend. |
| Tn_Representation | Enum | curve_linear | curve_linear, curve_rectangle, curve_bar, curve_point poles | Representation of each trend: curve_linear: point to point, direct curve_linear: point to point, vert/horiz curve_bar: vertical lines curve_point: points without connections poles: same as curve_bar. |
| Tn_Extrapolate | Enum | False | True, False | If the Extrapolate property is set to true, the curve will be extended up to current time, based on the last entered value. |
| Tn_Filter | Numeric | 0.0 | 0.0 to 1.0 | Filter factor: 0.0 : no filtration 1.0 : full filtration |
| Tn_Linewidth | Numeric | 1 | | Width of the draw-line for each trend. |
| Tn_Visible | Enum | True | True,False | Determines if trend number <i>n</i> and its Y-labels are visible. |

Table 41. Trend Properties

| Property | Type | Default | Range | Comment |
|-------------------------------|---------|---------|--|---|
| Tn_CurrentValue | String | *** | | Latest logged value for each trend (Read only). |
| Tn_MeanValue | String | *** | | Mean value for each trend (Read only). |
| Tn_RulerValue | String | *** | | Ruler value for each trend (Read only). |
| Tn_RulerTime | String | *** | | Ruler time for each trend (Read only). |
| Tn_Network | Numeric | | | Network number for the logged object. Currently not used. |
| Tn_Node | Numeric | | | Node number for the logged object. Used to specify a remote data provider. |
| Tn_Treatment | String | any | mean,max,min, inst,sum,sqsum, stcldev,numval, any | For trend <i>n</i> , lets you retrieve a log that performs a specific calculation. Applicable for ALM and LOGMGR DataSource . |
| Tn_TimeOffset | Numeric | 0 | | Time to delay a trend in seconds. |

The following sections provide guidelines for configuring Trend properties. Refer also to [Basic Properties](#) on page 218.

Scope

The following set of properties define the time range for the trends in the trend display element:

- Scope - This is the time period. The format is hh:mm:ss.
- ScopeStartDate & ScopeEndDate - These dates must be entered in the format specified in the TimeScaleDateFormat property. Default is: %Y-%m-%d, e.g. 1996-12-31

- **ScopeStartTime & ScopeEndTime** - These times must be entered in the format specified in property **TimeScaleTimeFormat**. Default is: %H:%M, for example: 16:45

Generally, the scope is defined by setting the **Scope**, **ScopeEndDate**, and **ScopeEndTime** properties. The **ScopeStartDate** and **ScopeStartTime** can be calculated from these settings. When switched to Run mode the trend element is always at current time. To change the scope to other times, the time scope properties must be set from another object (for example, a Button) and switch to the new scope, for instance by using zoom:

```
$Trend_1.ScopeStartDate = "1994/01/01";
$Trend_1.ScopeEndDate = "1994/01/01";
$Trend_1.ScopeStartTime = "12:00:00";
$Trend_1.ScopeEndTime = "14:00:00";
&Trend_1.Scope = 2:00:00;
$Trend_1.TrendMode = "ZoomTimeTime";
update("Trend_1");
```

DataSource

This property specifies the **DataSource** as **LOGMGR**, **SQL**, **OPCHDA**, or **ALM**. This specification is used for all trend curves in the Trend Display element as long as the **Tn_DataSource** property for each trend curve is set to **COMMON** (default). You can set the data source type for individual trend curves by setting the corresponding **Tn_DataSource** property to **LOGMGR**, **SQL**, **OPCHDA**, or **ALM**.

OPCHDA

This lets you access history data from an OPC server connected via the **OPCHDA** data provider specified in the **Tn_Logname** attribute:

- **AIPHDA** connects to the 800xA system OPC HDA server. This supports seamless access to operator trend logs and History Services history logs. It also supports access to log attributes. It does not support updates to history logs.
- **IMHDA** connects to the History Services OPC HDA server. This can only access history logs. It does not support access to log attributes. You must use **IMHDA** if you need to update history logs (add new entries or modify existing entries).

In addition to the `DataSource` and `Tn_logname` properties, you must also set the `Scope` and `Tn_ObjectName` properties.

The `Scope` value is applied to all trends. The `ScopeEndTime` (time on right side) is always current time, unless using some of the functions described later. The `ScopeStartTime` is calculated as the `ScopeEndTime` minus the `Scope` (unless you use `ZoomTimeTime`).

The `Tn_ObjectName` property is specified on an individual basis for each trend in the Trend display element. If you do not know the log name, if you want to be sure to enter it correctly, it is recommended that you use the OPC browser to find the log, and then copy and paste the log into the `Tn_objectName` property. For instructions on using the OPC Browser, refer to *System 800xA Information Management Data Access and Reports (3BUF001094*)*.

LOGMGR

To retrieve data from the Log Manager object set the `DataSource` to `LOGMGR`. This setting applies to all trends in the trend display element having their `Tn_DataSource` set to `COMMON`. This setting lets you specify data sources for trends using log access names.

In addition to the `DataSource` property, you must also set the `Scope` and `Tn_ObjectName` properties (`Tn_Attribute` and `Tn_Treatment` properties are not used for `LOGMGR` access).

The `Scope` value is applied to all trends. The `ScopeEndTime` (time on right side) is always current time, unless using some of the functions described later. The `ScopeStartTime` is calculated as the `ScopeEndTime` minus the `Scope` (unless you use `ZoomTimeTime`).

The `Tn_ObjectName` property is specified on an individual basis for each trend in the Trend display element. For example: `B0501,VALUE` or `TC101,MEASURE`.

There are several methods to specify `Tn_ObjectName`. For instance, you can:

- specify the object name directly in the `Tn_ObjectName` property, [Figure 97](#). With this method, if the name contains a hyphen (-) or period (.), the entire text string must be bounded by double-quotes. For example: “SC5_1-PID,MEASURE”

Assigning a value to Tn_ObjectName ——— `"SC5_1-PID,MEASURE"`

Figure 97. Explicit Specification of Object name

- reference another display element property such as Edit_1.Value, [Figure 98](#). When you use this method, when you enter the name in the edit field, DO NOT use double quotes, even if the name contains a hyphen (-) or period (.).

Assigning a value to Tn_ObjectName ——— `$Edit_1.Value`

Figure 98. Reference to Another Display Element Property

- assign a value to Tn_ObjectName from a script. For instance, you can write a script for a Button Action, [Figure 99](#). When you use this method, when you enter the name in the edit field, YOU MUST use double quotes, when the name contains a hyphen (-) or period (.). For example: "SC5_1-PID,MEASURE".

Script for Button Action ——— `$Trend_1.T1_ObjectName=$Edit_1.Value;
update ("Trend_1");`

Figure 99. Using a Script to Assign Tn_ObjectName

When the above information is entered, and the display is put in the Run mode, the trend display element should be displayed. If not, make sure:

- Tn_Visible property is set to true.
- the Tn_Color is visible on the TrendBackGround.
- the display element name is entered correctly.
- the specified data source has a log configured for it.

The Trend display element shows a maximum of 500 values, even though there may be more values for the specified scope. For better resolution of a specific area of the trend, you can use one of the zoom functions described later, or specify a smaller Scope.

SQL

To retrieve data from the Oracle database, set the DataSource to SQL. Enter the SQL query for each trend in Tn_Logname.

You must ensure that the query only requests data inside the selected (visible) area of the trend. The general form of the query should be:

```
"select to_char(<TIME_TAG>,'<FORMAT>') ,<VALUE>
from <TABLE_NAME>
where to_char(<TIME_TAG>,'<FORMAT>') >= `
+${<TREND_NAME>.ScopeStartDate + " "
+${<TREND_NAME>.ScopeStartTime+" '
and to_char(<TIME_TAG>,'<FORMAT>') <= `
+${<TREND_NAME>.ScopeEndDate + " "
+${<TREND_NAME>.ScopeEndTime+" ' "

```

where:

<TIME_TAG> is the date column

<FORMAT > is the Oracle date format string that converts TIME_TAG to the Time & Date format used in the trend element

<VALUE> is the value column in the database.

<TABLE_NAME>is the name of the table in the Oracle database

<TREND_NAME>is the name of the trend element itself.

Example:

Query data from a table called 'MY_DATA' with the following description:

| Name | Null? | Type |
|-------------|-------|------------|
| MY_TIME_TAG | | DATE |
| MY_VALUE | | NUMBER(38) |

The trend element has the settings:

```
Name = MyTrend
TimeScaleTimeFormat: %H:%M:%S
TimeScaleDateFormat: %Y/%m/%d
```

The query to enter in the Tn_LogName is:

```
"select to_char(MY_TIME_TAG, 'YYYY/MM/DD
HH24:MI:SS') , MY_VALUE
from MY_DATA
where to_char(MY_TIME_TAG, 'YYYY/MM/DD HH24:MI:SS') >= '
+$MyTrend.ScopeStartDate + " " + $MyTrend.ScopeStartTime + "'
and to_char(MY_TIME_TAG, 'YYYY/MM/DD HH24:MI:SS') <= '
+$MyTrend.ScopeEndDate + " " + $MyTrend.ScopeEndTime + "' "
```

This ensures that only data inside the trend area are requested. You can use additional WHERE clauses in the statement as required.

ALM (Non-standard Master applications only)

To retrieve data from the ALM, set the DataSource to ALM. Then follow the instructions for [LOGMGR](#). When you use ALM access, in addition to specifying Tn_ObjectName for each trend, you must also specify:

- Tn_Attribute (MV, MEASURE, or any other trended attribute)
- Tn_Treatment (MEAN or AVERAGE)

Channel Number

For each trend (T1 to T6) you can specify the channel number for a remote data provider. This lets you display trend data from more than one location (from different display servers) on the same trend graph. How to install and configure remote data providers is described in the section on data providers in *System 800xA Information Management Configuration (3BUF001092*)*. If you do not use remote data providers, then use the default channel number (0).

Ruler & RulerBehaviour

When a trend display is in the Run mode, you can click in the trend area to display a vertical ruler. The ruler is positioned according to the RulerBehaviour property (AID or OS).

When RulerBehaviour is AID, when you click in the trend area, the ruler is placed at the nearest value within the scope for the selected trend curve. The RulerValue and RulerTime will contain the values for the selected trend corresponding to the selected time. The RulerValue and RulerTime for the other trends will contain values only if data was found for this exact time, otherwise they will contain '***'. If the selected trend has no data within the scope, the ruler will not be displayed.

Also, when RulerBehaviour is AID, the RulerStep property lets you move the ruler the specified number of steps of data. For example, RulerStep = 3, moves the ruler to the third time/value entry greater than the current one. RulerStep = -2, moves the ruler to the second time/value entry less than the current one.

When RulerBehaviour is OS, when you click in the trend area, the ruler is placed at the cursor location whether there are data for the selected trend or not. The RulerValue and RulerTime will contain the values of the nearest time LESS than the time of the ruler, for all trends having data.

Also, when RulerBehaviour is OS, the RulerStep property lets you move the ruler the specified percentage of the current time scope. For example, RulerStep = 10, moves the ruler ten percent to the right. RulerStep = -25, moves the ruler twenty five percent to the left.

To select a trend, set the SelectedTrend property to the number of the trend.

When the ruler is drawn, the *Tn_RulerTime* and *Tn_RulerValue* properties are updated for each trend. The RulerAction is executed at the same time that the ruler is positioned. This provides a means of forcing an update of other display elements.

To move the ruler, click another spot inside the draw-area. As the trends are updated with new data, the ruler will move to the left of the draw area, until it reaches the left side and disappears.

You can use another display element to force the ruler to move by specifying a RulerStep value and updating the Trend display element. A negative rulerstep value moves the ruler left, and a positive RulerStep value moves the ruler right.

Example:

Move the ruler in Trend_1 four steps to the left where RulerBehaviour = AID:

```
$Trend_1.RulerStep = "-4";  
update(Trend_1);  
execute(Trend_1,RulerAction); // if needed to update other display  
elements
```

After the ruler has been moved, the RulerStep property is automatically set to zero.

If no ruler appears, make sure the proper trend is selected, and the color of the ruler is visible on the specified TrendBackground.

With the RulerTimeFormat property, you can control the format of the RulerTime property. For example, RulerTimeFormat = %Y/%m/%d %H:%M:%S, will result in a time formatted like: 1996/08/31 14:35:17.

The RulerValue property includes a data status indicator:

- No indicator means the data is OK
- A question mark (?) after the data means NoData
- an exclamation mark (!) after the data means BadData

TrendMode

The TrendMode property can be used to make the Trend display element perform an action initiated by another display element. The modes are:

[Normal](#)

[UpdateTraces](#)

[ZoomInOut](#), [ZoomArea](#), [ZoomTimeDuration](#), [ZoomTimeTime](#)

[ChangeScope](#)

[GetLoggedAttributes](#)

[ClearLogData](#)

[QueryValues](#), [QueryValuesPlusYMinMax](#), [QuerySelected](#), [QueryPlusYSelected](#)

[ResetYValuesSelected](#).

The actions are only used in run-mode. To invoke an action, set the TrendMode property from another display element and then update the Trend display element. For example:

```
$Trend_1.TrendMode = "ClearLogData";  
update("Trend_1");
```

If the Trend display element should start to query values immediately after entering the display, the last line in the OnPreQuery property of the display should set TrendMode to one of the modes that queries values. For example:

```
$Trend_1.TrendMode = "QueryValues";
```

The update statement is not needed here since queries are always executed upon entering a display.

Normal

This is the default mode. No actions are executed when TrendMode = Normal.

UpdateTraces

This mode performs the following actions:

- unlocks the display after Zoom or Scope change.
- hides the Ruler, if it was placed by the user.
- sets the ScopeEndTime to current time and uses the current scope.
- sets TrendMode to Normal and clears the internal data buffer.
- sets CurrentValue and MeanValue to ***.
- queries values for all trends.

Example: Set the timescope to one hour, and the stoptime to current time.

```
$Trend_1.Scope = "1:00:00";  
&Trend_1.TrendMode = "UpdateTraces";  
update("Trend_1");
```

ZoomInOut

This mode zooms in or out symmetrically around the ruler based on a specified zoom rate. If no ruler is present, the ruler is positioned in the center of the curve area. The zoom rate is specified in the ZoomPercent property as a percentage of the scope. For example, if the current scope is one hour, setting ZoomPercent = 80 will result in a new scope of 48 minutes (80% of one hour). Numbers less than 100 will

zoom in, and numbers larger than 100 will zoom out. After performing the Zoom operation, the ZoomPercent property is set to zero.

Example: Reduce the timescope to 80% of the current scope.

```
$Trend_1.TrendMode = "ZoomInOut";  
&Trend_1.ZoomPercent = 80;  
update("Trend_1");
```

The display is locked after the scope is changed. When the DataSource is [ALM](#), this is indicated by a border around the draw area (the color of the border is specified by the ScopeLockColor property). When the display is locked, the time scale is not updated, though the trends are updated as new data are received. To unlock the display, use the UpdateTraces mode. Zooming is only allowed if the resulting timespan is greater than one minute.

ZoomArea

This mode zooms inside an area specified by two markers. To specify the zoom area, first set the TrendMode to ZoomArea and update the Trend display element.

When the Trend display element is set to ZoomArea, the display is locked. Click inside the draw area twice - first to mark the starting point, and again to mark the ending point of the area you wish to zoom. The marked area will fill the entire scope, and the Trend display element will be in locked mode. To perform another zoom area operation, repeat this procedure.

If the chosen zoom area is less than one minute, the ScopeEndTime is set to the chosen zoom stop time, and the Scope property is set to one minute.

Example: Activate the ZoomArea function:

```
$Trend_1.TrendMode = "ZoomArea";  
update("Trend_1");
```

After this, place the ruler inside the trend area twice to mark the start and end points, then the zooming will be performed.

ZoomTimeDuration

This mode lets you specify a new scope for the trend display element directly using:

- ScopeEndTime (in TimeScaleTimeFormat)
- ScopeEndDate (in TimeScaleDateFormat)
- Scope (in seconds)

Once these properties have been set, set the TrendMode property to ZoomTimeDuration and update the Trend display element.

Example: Show trends from the 31st of december 1993 in a 2 hour scope from two o'clock.

```
$Trend_1.ScopeEndDate = "1993/12/31";  
$Trend_1.ScopeEndTime = "14:00:00";  
$Trend_1.Scope = "2:00:00";  
$Trend_1.TrendMode = "ZoomTimeDuration";  
update("Trend_1");
```

If no scope parameters are set, the current values are used.

ZoomTimeTime

This mode is similar to ZoomTimeDuration, except that you specify start time & date and end time & date rather than scope.

Example: Show the same trends as the example in ZoomTimeDuration.

```
$Trend_1.ScopeEndDate = "1993/12/31";  
$Trend_1.ScopeEndTime = "14:00:00";  
$Trend_1.ScopeStartDate = "1993/12/31";  
$Trend_1.ScopeStartTime = "12:00:00";  
$Trend_1.TrendMode = "ZoomTimeTime";  
update("Trend_1");
```

ChangeScope

The ChangeScope mode changes both start time and end time equally to pan the draw-area. To modify the scope, use the ScopeAdjustment property to specify how much to change the start time and end time with respect to the scope. A negative number moves the scope left and a positive number moves the scope right.

Example: A trend has starttime 12:00 and endtime 13:00. To change the starttime to 11:30 and the endtime to 12:30, and lock the display:

```
$Trend_1.ScopeAdjustment = "-0.5";  
$Trend_1.TrendMode = "ChangeScope";  
update("Trend_1");
```

GetLoggedAttributes

This mode is only applicable for [ALM](#) access. This mode retrieves the logged attributes for a given display element from ALM. The number of logged attributes of an object is between 0 and 7. To perform the GetLoggedAttributes, you must specify the ObjectName in Tn_LogName and the 'Treatment' in Tn_Treatment. Once these properties have been defined, set TrendMode to 'GetLoggedAttributes' and then update the Trend display element.



Since answers are received asynchronously, the values are placed in the captions of seven Buttons named 'SetLoggedAttribute_[1-7]' which must be created before executing the query.

When all answers are placed at each of the seven captions, every Button is executed from the Trend display element. This lets you place the retrieved information where it should be used.

ClearLogData

This mode clears the internal buffer of the selected trend curve, unlocks the trend, and sets the TrendMode to Normal.

QueryValues

This mode queries values for all trends, depending on the DataSource specification.

- If the DataSource is [LOGMGR](#) this mode lets you update trends after you change the Scope or Tn_Objectname. The actions performed are:
 - Unlock the trend, if it is locked
 - Clear the internal buffers for the trend
 - Query values from the Log Manager

- If the DataSource is [ALM](#), this mode lets you update trends after you change the Scope or Tn_Objectname. The following actions are performed:
 - Unlock the Trend, if it was locked
 - Clear the internal buffers of all trends
 - Query values from ALM
 - If properties Scalemin and Scalemax are equal, the following are read from ALM: Ymax, Ymin, unit and description. If properties Scalemin and Scalemax are not equal, just unit and description are read from ALM (not Ymin and Ymax).
- If the DataSource is [SQL](#) this mode lets you query current-time data. The following actions are performed:
 - Clear the internal buffers of all the trends
 - Query values from Oracle for the trends.

QueryValuesPlusYMinMax

This mode is similar to QueryValues, except for the following. For ALM, this mode always requests Y-Min/Max values from the ALM and places them in Tn_Scalemax and Tn_Scalemin respectively. The previous setting of Tn_Scalemax and Tn_Scalemin are ignored.

QuerySelected

This mode operates similar to QueryValues, except that it queries values for the selected trend only, rather than for all trends.

QueryPlusYSelected

This mode operates very similar to QueryValuesPlusYMinMax, except that it queries values for the selected trend only, rather than for all trends.

ResetYValuesSelected

This applies to ALM-mode only. It requests the following for the selected trend only: unit, description and Y-Min/Max. After this, the TrendMode is set to Normal.

Representation

Tn_Representation determines how the trends are drawn. The options are:

- `curve_linear` - Values are connected directly from point to point, drawn with the specified line width, [Figure 100](#).
- `curve_rectangle` - Values are connected horizontally and vertically, drawn with the specified line width, [Figure 101](#).
- `curve_bar` & poles - These are the same because of backwards compatibility. Values are not connected, but shown as vertical poles drawn from zero to the current value, drawn with the specified line width, [Figure 102](#).
- `curve_point` - Data are drawn as circular points, with the line width property used as the diameter, [Figure 103](#).

Examples of the different drawing methods is shown in [Bad/No data Color](#) on page 192.

Bad/No data Color

When trend curves are drawn, data status is examined and the curve (whole or part) is color coded to reflect the status. The NoDataColor and BadDataColor properties determine what colors are used to indicate NoData and BadData respectively. Good data are still drawn using the Tn_Color property. The properties are common for all curves, and the function is illustrated in [Figure 100](#) through [Figure 103](#).

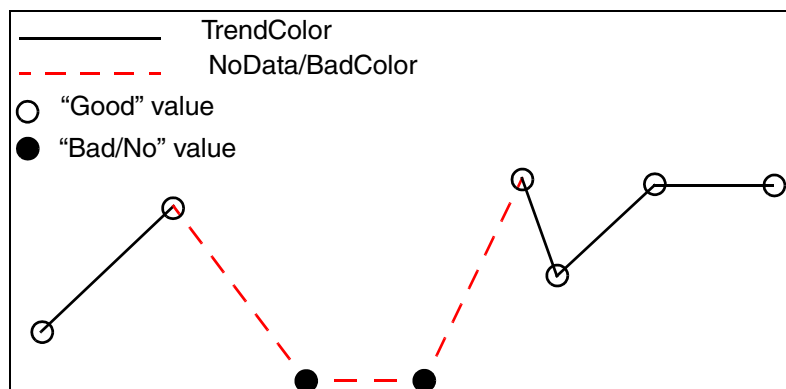


Figure 100. Curve Drawn in Curve_Linear Mode

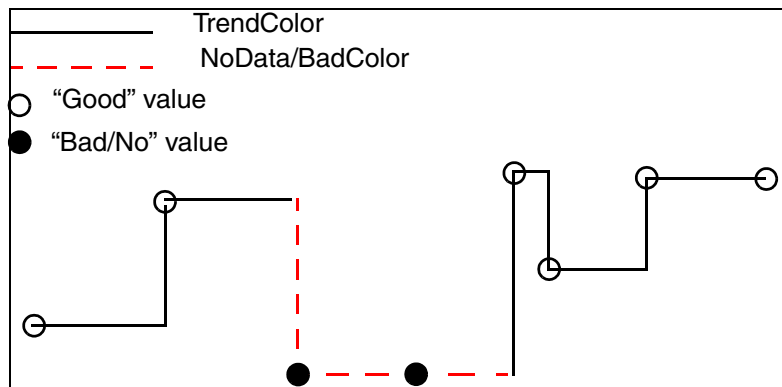


Figure 101. Curve Drawn in Curve_Rectangle Mode

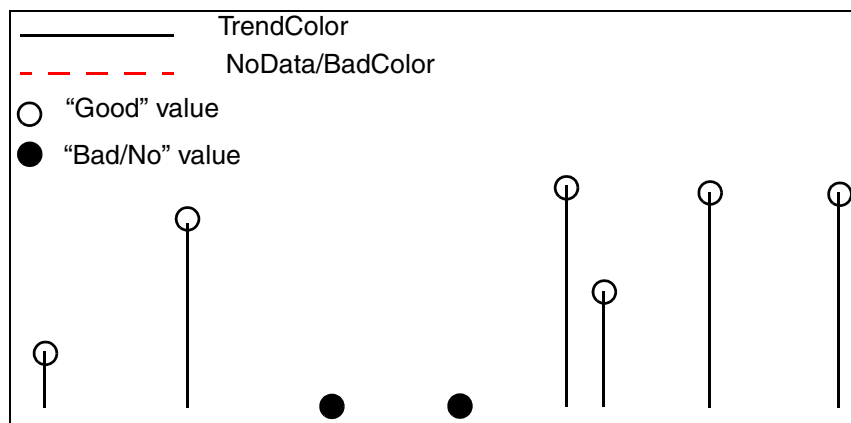


Figure 102. Curve Drawn in Poles or Curve_Bar Mode

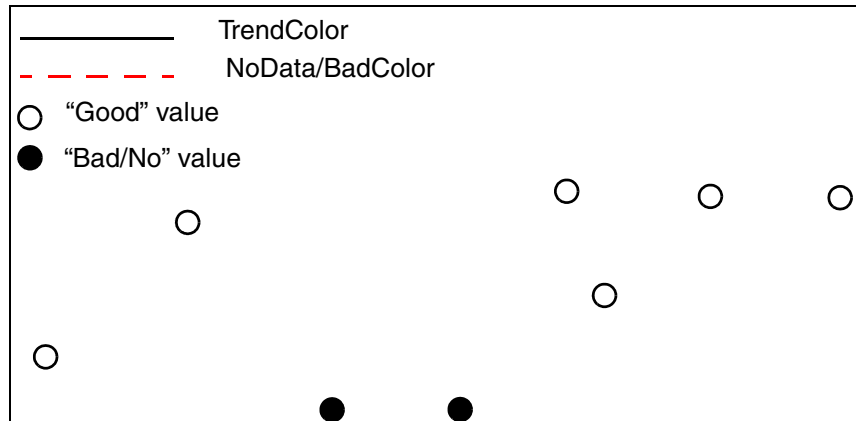


Figure 103. Curve Drawn in Curve_Point Mode

GapToEdge

When the trend displays data from current time and backward, a gap is shown right to the curve. The gap width is controlled by the GapToEdge property. This property specifies the gap width in percent of the trend drawing area. The value must be in the interval 1-10%.

Extrapolate

When a curve is based on an asynchronous log, data is often missing at current time. If the `Tn_Extrapolate` property is set true, the curve will be extended up to current time, based on the last entered value. This is shown in [Figure 104](#).

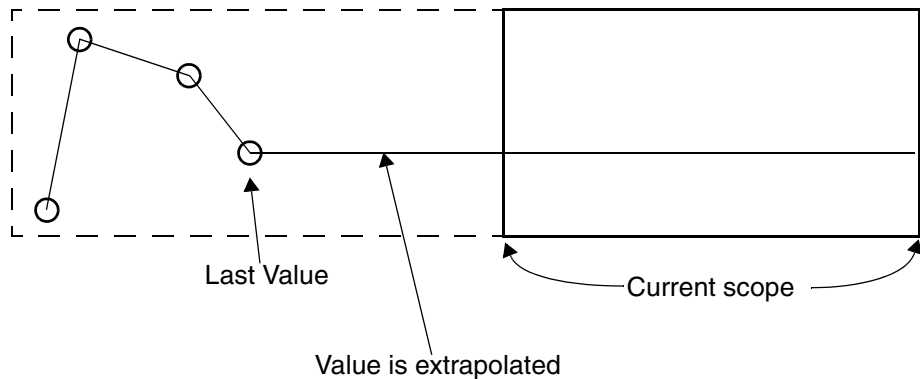


Figure 104. Curve Drawn in Curve_Linear Mode with Extrapolate = True

Filter

You can apply a filter function to a single trend before drawing it. The filter reduces jitter from trends that oscillate. The filter factor must be between 0 and 1, where 0 means no filter and 1 is 100 percent filter (resulting in a horizontal line).

The filter is only applied to the part of the trend which is visible. You can change the filter factor in Run mode.

Examples of filter factors (0, 0.5, 0.75 and 0.95 from top to bottom) are shown in [Figure 105](#).



Figure 105. Example Filter Factors

TimeOffset

The timeoffset lets you compare one location (time) on a trend to another location on the same trend. The TimeOffset value is specified in seconds. The `Tn_TimeOffset` is added to each `time_tag` for the trend. A positive value moves the trend to the right in the draw area.

An example is shown in [Figure 106](#). Two trends display the same data. To compare the value at 9:12:30 in the upper trend, with the value at 9:10:00 in the lower trend, set the TimeOffset property in the lower trend to 2:30. The result is shown in the lower (After Offset is Applied) view. Placing the Ruler at 9:12:30 places the value of 9:12:30 in the RulerValue property of the upper trend, and the value of 9:10:00 in the RulerValue property of the lower trend.

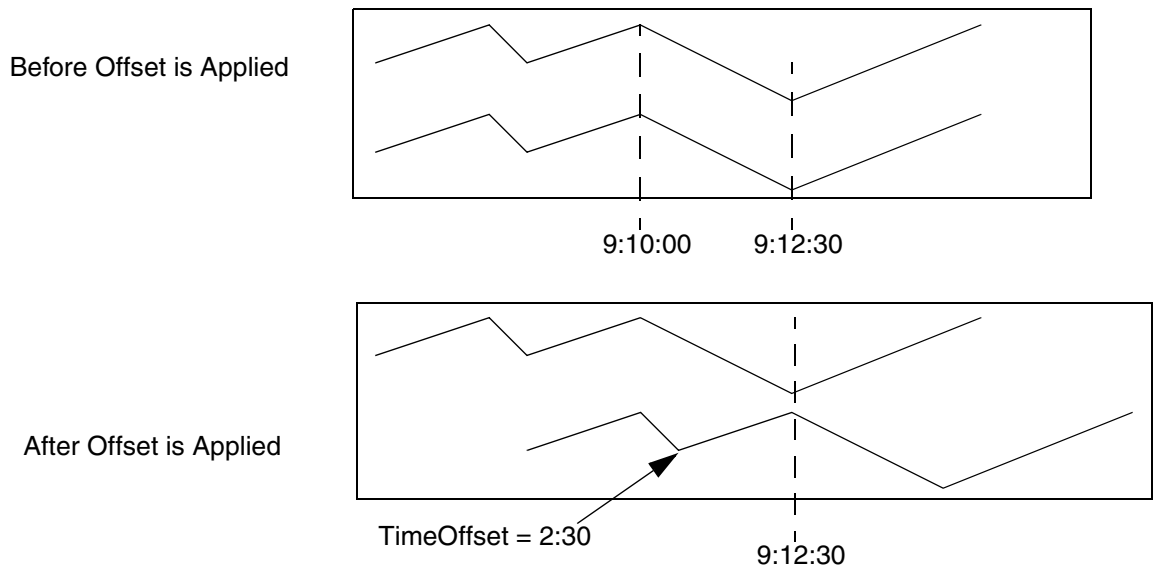


Figure 106. Example, TimeOffset



TimeOffset can be used to compare trends that are not related by time. For instance, you can use it to compare cyclic events such as batches, or coordinated process variables with known creation or lag times.

Gauge

The toolbar button for the Gauge display element is shown in [Figure 107](#).

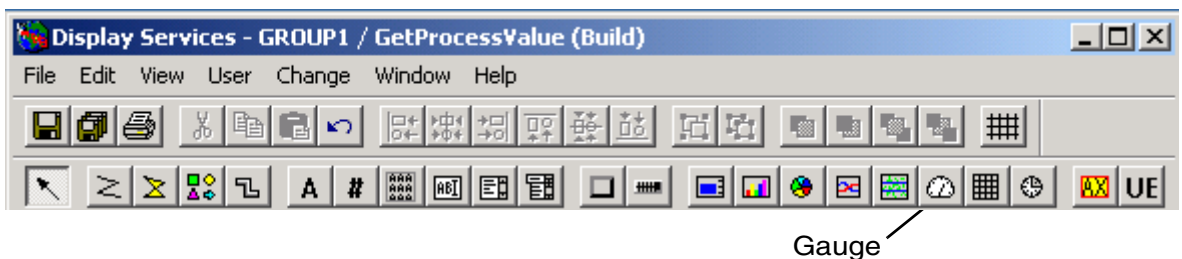


Figure 107. Gauge Button

The Gauge displays values in a meter-like fashion, [Figure 108](#). The needle indicates the current result of the dataquery. Configure the [Gauge properties](#), [Table 42](#), to set behavior and appearance characteristics.

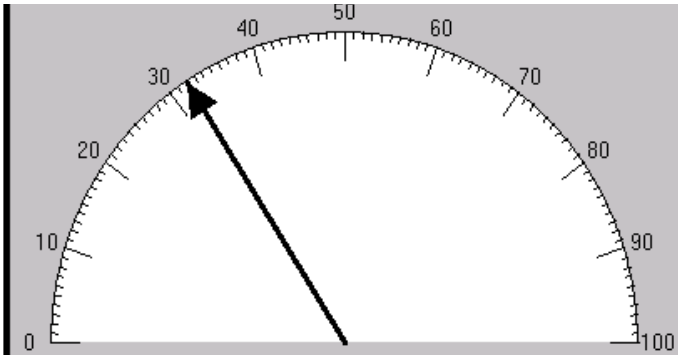


Figure 108. Example, Gauge Display Element

Table 42. Gauge Properties

| Property | Type | Default | Range | Comment |
|----------------------------|-----------------------|------------|------------|--|
| Name | String | Gauge_1 | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment to be displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not gauge is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of gauge frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of gauge frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the gauge frame. |
| Height | Integer | As created | 5 to 1200 | Height of the gauge frame. |
| Foreground | Color | 0 | -1 to 127 | Color of border. |

Table 42. Gauge Properties

| Property | Type | Default | Range | Comment |
|-----------------|----------|---------|--|---|
| Background | Color | 16 | -1 to 127 | Background color of the gauge. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |
| Framestate | Enum | In | In,Out | used to show depth. |
| Update | Enum | Demand | Demand, Cyclic_3s, Cyclic_6s, Cyclic_9s, Cyclic_15s, Cyclic_27s, Cyclic_30s, Cyclic_1m, Cyclic_5m, Cyclic_10m, Cyclic_15m, Cyclic_30m, Cyclic_1h | Update mode and frequency for SQL-based DataQuery. Demand = update on ReQuery. |
| DataQuery | Query | | | Script for data retrieval. |
| GaugeLocation | Enum | Bottom | Left,Top,Right,Bottom | Location of the needle rotation point. |
| GaugeColor | Color | 18 | -1 to 127 | Color of the arc area. |
| MajorTickNumber | Integer | 0 | 0 to 10 | Number of major ticks between max. and min. |
| MinorTickNumber | Integer | 0 | 0 to 10 | Number of minor ticks between two major ticks. |
| Scale | Enum | False | True,False | Controls the visibility of the scale text |
| Scalemax | Float | 100 | | |
| Scalemin | Float | 0 | | |
| ScaleFormat | Textline | %.0f | | Format of scale presentation. |

Table 42. Gauge Properties

| Property | Type | Default | Range | Comment |
|--------------|---------|---------|---|--|
| ScaleFont | Font | 3 | 0 to 71 | Font size for scale text. |
| ScaleColor | Color | 0 | -1 to 127 | Color of the scale text. |
| NeedleType | Enum | Line | Line, ArrowToCenter ArrowFromCenter, | The shape of the needle. |
| NeedleColor | Color | 0 | -1 to 127 | Color of the needle. |
| NeedleWidth | Integer | 1 | | The width of the needle. |
| NeedleLength | Integer | 100 | 0 to 100 | Needle length as percent of needle max. length. |

The following sections provide guidelines for configuring Gauge properties. Refer also to [Basic Properties](#) on page 218.

GaugeLocation & -Color

GaugeLocation determines the position of the needle’s rotation point. The options are Left, Top, Right and Bottom. In [Figure 108](#), the GaugeLocation is set to Bottom. GaugeColor determines the color of the arc area. You can use any color except transparent.

Major- & MinorTickNumber

The TickNumber properties determine the number of ticks (value-marks) on the border of the arc. The ticks are drawn in the foreground color.

The MajorTickNumber is the number of major ticks between the minimum and maximum value. The MinorTickNumber is the number of minor ticks between two major ticks. In [Figure 108](#), MajorTickNumber is set to 3, and MinorTickNumber is set to 2. If MajorTickNumber is set to 0, neither major nor minor ticks will be displayed.

Scale

Scale determines whether or not to display scale values for major tick marks. If the Scale property is true, a numeric scale value is displayed at each major tick. If MajorTick is set to 0, a scale value is displayed only at ScaleMin and ScaleMax.

Scale values are calculated from the values you specify for ScaleMax and ScaleMin. The scale values are presented in the specified ScaleFormat, and drawn in the specified ScaleColor.

The default position of the ScaleMax value depends on the GaugeLocation setting. The possibilities are shown in [Figure 109](#).

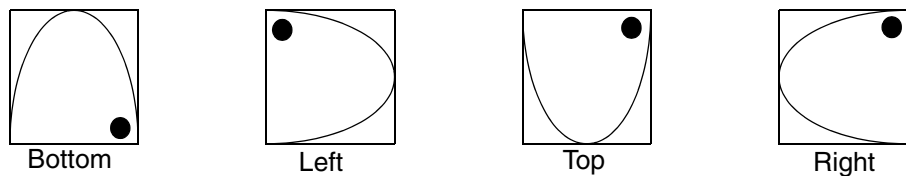


Figure 109. Default ScaleMax Locations

You can reverse the location of ScaleMax and ScaleMin values by entering the ScaleMax value in ScaleMin, and the ScaleMin value in ScaleMax.

Needle

NeedleType can be specified as: Line, ArrowFromCenter or ArrowToCenter. In [Figure 108](#), the NeedleType is ArrowFromCenter.

NeedleColor is the color of the needle, and NeedleWidth is the width. NeedleLength is the length of the needle in percent of the total length from the center to the border for a given value.

SmartShade

The toolbar button for the SmartShade display element is shown in [Figure 110](#).

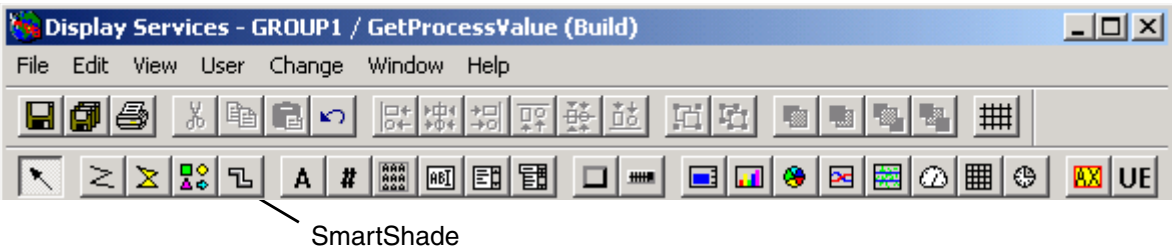


Figure 110. SmartShade Button

The SmartShade display element is used to draw 3D-like tanks, pipes and so on. SmartShades are drawn as if the light source is in the upper left corner of the screen, thus providing a 3D-like effect. The SmartShades can have different shapes as shown in [Figure 111](#).

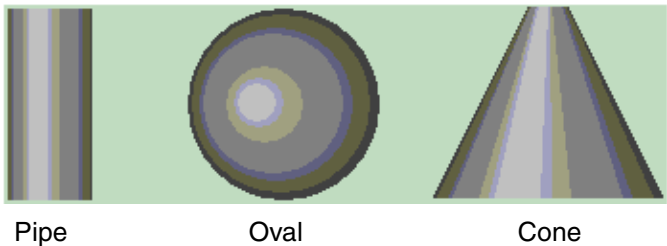


Figure 111. Different SmartShade Elements

The SmartShade element type has no data query functions. It is used only as a drawing element (like the Shape element). Each shape has its own set of properties that can be used to create various effects, [Figure 112](#).

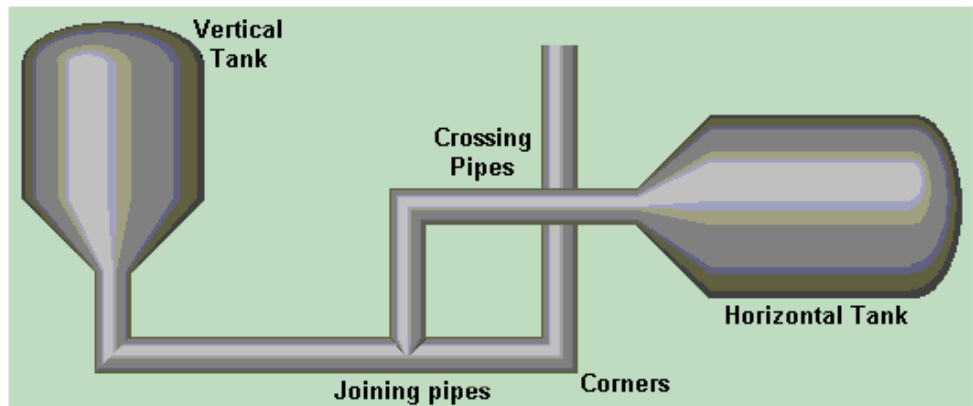


Figure 112. Example with SmartShades

Configure the [SmartShade properties](#), [Table 43](#), to set behavior and appearance characteristics.

The contents of the property list vary depending on the currently selected Shape property. This is illustrated in [Table 43](#) by shading the variable properties.

Table 43. SmartShade Properties

| Property | Type | Default | Range | Comment |
|-------------------------|---------|--------------|------------|--|
| Name | String | SmartShade_1 | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not SmartShade is visible in Run mode. |
| X | Integer | As created | 0 to 1200 | X coordinate of SmartShades frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of SmartShades frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of the SmartShade frame. |
| Height | Integer | As created | 5 to 1200 | Height of the SmartShade frame. |

Table 43. SmartShade Properties

| Property | Type | Default | Range | Comment |
|------------|---------|----------|----------------------------|--|
| Foreground | Color | 23 | -1 to 127 | Brightest color of the SmartShade. |
| Background | Color | 9 | -1 to 127 | Darkest color of the SmartShade. |
| Shape | Enum | Pipe | Pipe, Oval, Cone | The shape of the SmartShade. |
| Direction | Enum | Vertical | Vertical, Horizontal | Only when Shape = Pipe |
| TopEnd | Enum | Flat | Flat, Left, Right | Only when Shape = Pipe and Direction = Vertical. |
| BottomEnd | Enum | Flat | Flat, Left, Right | Only when Shape = Pipe and Direction = Vertical. |
| LeftEnd | Enum | Flat | Flat, Up, Down | Only when Shape = Pipe and Direction = Horizontal. |
| RightEnd | Enum | Flat | Flat, Up, Down | Only when Shape = Pipe and Direction = Horizontal. |
| Light | Enum | Vertical | Vertical, Horizontal | Only when Shape = Oval |
| Direction | Enum | Down | Down, Left Right, Up | Only when Shape = Cone |
| ConeSize | Integer | 0 | 0 - Width or 0 - Height | Only when Shape = Cone |

The following sections provide guidelines for configuring SmartShade properties. Refer also to [Basic Properties](#) on page 218.

For instructions on combining smart shade elements to achieve certain visual effects see [Transitions](#) on page 208 and [Joining Pipes](#) on page 208.

Foreground & Background

By default, all SmartShade elements consist of a blend of colors from the palette, ranging from and including 23 (Background) to 9 (Foreground). Background is used as the darkest color at the border of the element, and Foreground is used as the brightest color.

Although you can, it is strongly recommended that you DO NOT modify the default foreground and background colors. If you do, in order to gain the 3D-like effect, the range should be consecutive grey-scale colors. For instance, selecting a brighter Foreground color tending to white, will give the illusion of a sharper lightsource.

Pipe

The Pipe shape is used to draw pipes and tanks. The direction of the shape can be either vertical or horizontal, and depending on the direction the ends of the shape can be drawn either flat or with a bevel. The different options are illustrated in [Figure 113](#).

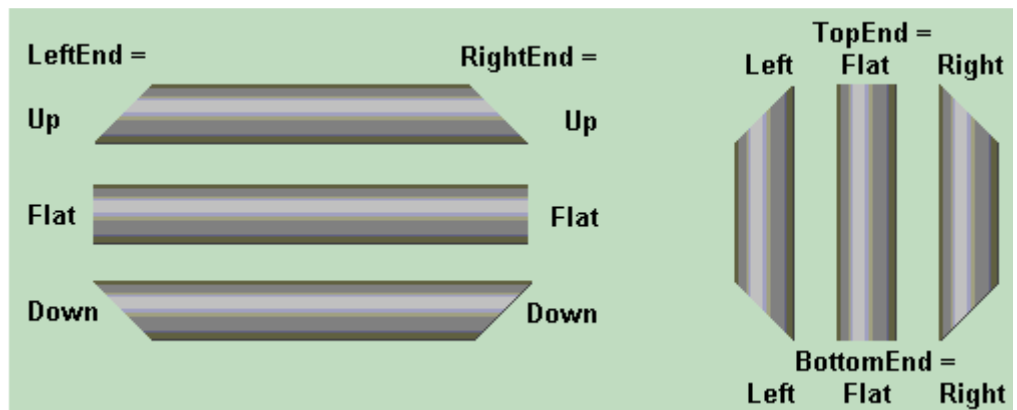


Figure 113. Pipe SmartShades

When connecting two pipes in bevel, the height of the horizontal pipe should match the width of the vertical pipe.

Oval

The Oval is mainly used to create round ends on tanks. When Shape is set to Oval, the Light property is included in the Properties List. It can be either Vertical or Horizontal, as shown in [Figure 114](#).

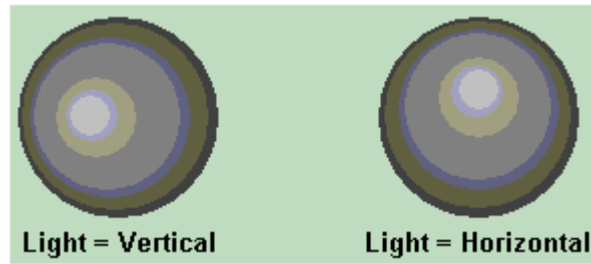


Figure 114. Oval SmartShades

This property is used to show the orientation of the tank as illustrated in [Figure 115](#).

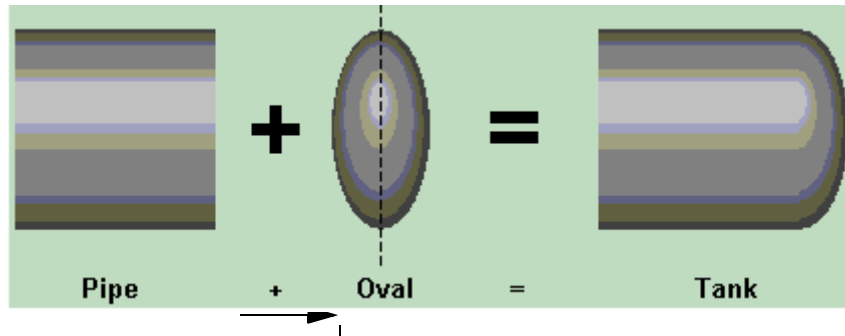


Figure 115. Creating tanks

When you create tanks like this, make sure the Ovals are behind the Pipes in the Z-order. This is done by selecting the Oval, and then choosing **Change > Order > MoveToBack** from the main menu bar. You can also change Z-order by choosing **Layout > Element List**.

The following are recommendations to help you match the different colors precisely when overlapping the objects:

- For horizontal tanks:

- The height of the Oval and Pipe should be equal.
- The width of the Oval should be half the height of the Oval.
- For vertical tanks:
 - The width of the Oval and Pipe should be equal.
 - The height of the Oval should be half of the width of the Oval.

Cone

The Cone is used to make transitions between, or join two Pipes with different widths. There are two properties associated with the Cone: Direction and ConeSize.

The Direction, [Figure 116](#), refers to the position of the smaller end of the Cone.

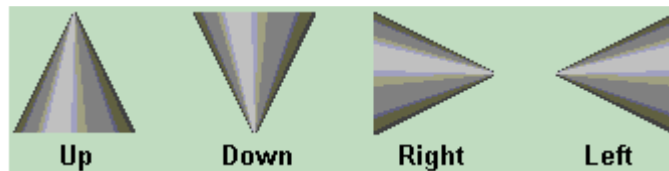


Figure 116. Cone SmartShade, Directions

ConeSize, [Figure 117](#), determines the actual width of the smaller end of the Cone.

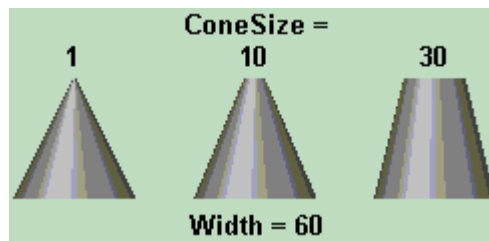


Figure 117. Cone SmartShade, ConeSize

The ConeSize property does not change, even if you change the Width of the Cone. Changing ConeSize will affect the overall appearance of the cone. For Directions Up & Down:

- If ConeSize is greater than zero and smaller than Width, the Cone is drawn normally.
- If ConeSize is less than or equal to zero, or ConeSize is greater than or equal to Width, the Cone is drawn as a vertical Pipe with the width = Width.

For Directions Left & Right:

- If ConeSize is greater than zero and smaller than Height, the Cone is drawn normally.
- If ConeSize is less than or equal to zero, or ConeSize is greater than or equal to Height, the Cone is drawn as a horizontal Pipe with the height = Height.

Transitions

To create a transition between two horizontal Pipes, the Cone must have the Height property set to the Height property of the larger Pipe, and ConeSize must match the height of the smaller Pipe, as illustrated in [Figure 118](#).



Figure 118. Cone SmartShade, Transitions

Joining Pipes

To create the illusion of joining Pipes simply let a Cone with ConeSize = 1 overlap the Pipe being joined. Depending on directions, the following applies:

- For joining a horizontal Pipe:
 - Direction of Cone: Up or Down
 - Height of Cone: Half the height of the horizontal Pipe
 - Width of Cone: Width of vertical Pipe.
- For joining a vertical Pipe:
 - Direction of Cone: Left or Right

- Width of Cone: Half the width of the vertical Pipe
- Height of Cone: Height of the horizontal Pipe

The [Figure 119](#) demonstrates some joinings made to a horizontal Pipe.

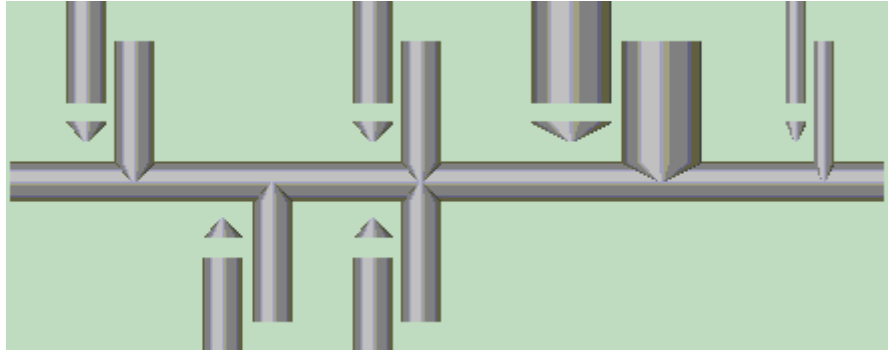


Figure 119. Cone SmartShade, Joinings

XYPlot

The toolbar button for the XYPlot display element is shown in [Figure 120](#).



Figure 120. XY Plot Button in Toolbar

The XYPlot element displays current data held in a Matrix element, [Figure 121](#). Each XYPlot display element can hold up to six curves. The element has features for viewing the plot curves such as zooming, placing a ruler, and changing the scope in run-time.

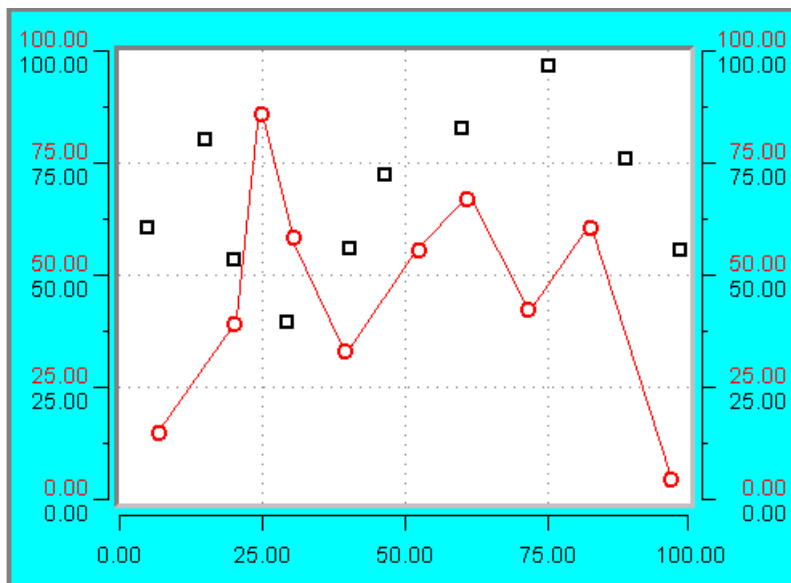


Figure 121. Example, XYPlot Display Element

The XYPlot element retrieves data from the matrix element. The XYPlot element can function as a stand-alone-display element; however to use the advanced functions mentioned above, it must interact with other elements and some scripting will be needed.

Configure the [XYPlot properties, Table](#) , to set behavior and appearance characteristics.

The XYPlot display element has a number of properties common to all plot curves, in order to set such characteristics as:

- font, color and visibility of the x-scale.
- color of the plot area.
- visibility and color of the dotted grid lines.
- visibility and color of the border.

Then there are several properties that apply to the individual plot curves:

- representation of each curve (scatter or curve).

- visibility, color, and linewidth.
- point size and display mode (circle, square or diamond).
- scale min/max and font.

Some properties are read-only, and are read by other display elements, to display such data as:

- current values for a plot curve.
- ruler x and y values.

Refer to [Guidelines for Building an XYPlot](#) on page 133 for details on how to coordinate interaction between the Matrix and XYPlot elements.

Some properties require scripting. Refer to [Section 6, Display Scripting](#).

Table 44. XYPlot Properties

| Property | Type | Default | Range | Comment |
|----------------------------|-----------------------|------------|--------------|---|
| Name | String | Type_# | | Name must be unique within the display |
| Comment | String | | 255 chars | Optional comment to be displayed in Run mode. |
| Visible | Enum | True | True, False | Whether or not plot is visible in run mode |
| X | Integer | As created | 0 to 1200 | X coordinate of the XYPlot frame's upper left corner. |
| Y | Integer | As created | 0 to 1200 | Y coordinate of the XYPlot frame's upper left corner |
| Width | Integer | As created | 5 to 1200 | Width of the XYPlot frame. |
| Height | Integer | As created | 5 to 1200 | Height of the XYPlot frame. |
| Foreground | Color | 0 | -1 to 127 | Color of border |
| Background | Color | 16 | -1 to 127 | Background color of the XYPlot. |
| Framewidth | Integer | 3 | 0 to Width/2 | Width of the Motif frame. |

Table 44. XYPlot Properties

| Property | Type | Default | Range | Comment |
|----------------|---------|---------|---------------|---|
| Framestate | Enum | In | In, Out | Used to show depth |
| Update | Enum | Yes | Yes, No | Update mode |
| PlotAreaColour | Color | 23 | -1 to 127 | Color of the curve area. Default is black. |
| VerticalGrid | Enum | True | True, False | If True, the vertical grid lines are visible. |
| HorizontalGrid | Enum | True | True, False | If True, the horizontal grid lines are visible. |
| GridColor | Color | 31 | -1 to 127 | Color of the grid lines. Default is grey. |
| ShowXLabels | Enum | True | True, False | Show/hide X-ruler and labels |
| ShowYLabels | Enum | True | True, False | Show/hide Y-ruler and labels |
| RulerColor | Color | 31 | -1 to 127 | Color of the ruler |
| RulerAction | String | | | Script for action executed each time the ruler is placed in the curve area. Typically used to update other display elements |
| XScaleColor | Color | 31 | -1 to 127 | Color of the X-scale |
| XScaleFormat | String | %.2f | | The format of the X-scale marks |
| XScaleFont | Font | 1 | 1 to 71 | Font for the X-scale marks |
| XScaleMax | Numeric | 100 | integer range | Max shown x-value for XYPlot. |
| XScaleMin | Numeric | 0 | integer range | Min shown x-value for XYPlot. |
| XRulerValue | Float | None | float range | Ruler position along X-axis |
| RulerVisible | Enum | False | True, False | Is the ruler visible |

Table 44. XYPlot Properties

| Property | Type | Default | Range | Comment |
|---|-----------------------|-------------------------------|---------------------------|--|
| Pn_Xserie⁽¹⁾ | Query | none | none | Script to retrieve X-data for one curve from a Matrix element. |
| Pn_Yserie | Query | none | none | Script to retrieve Y-data for one curve from a Matrix element. |
| Pn_Representation | Enum | Curve | Scatter, Curve | Curve display mode |
| Pn_ScatterType | Enum | Circle | Circle, Square or Diamond | Point display mode. Default is Circle. |
| Pn_ScatterStrength | Numeric | 1 | 1 to 6 | Pen width for the plot marks. Default is 1. |
| Pn_LineWidth | Numeric | 1 | 0 to 8 | Pen width for the curve. |
| Pn_PlotColor | Color | | -1 to 127 | Color of the XY-Plot |
| Pn_LimLinesVisible | Enum | False | True, False | If True, the limit lines are visible |
| Pn_LimitH | Float | 0.0 | float range | Upper limit line position |
| Pn_LimitC | Float | 0.0 | float range | Center line position |
| Pn_LimitL | Float | 0.0 | float range | Lower limit line position |
| Pn_HLimColor | Color | Pn_Plot-Color | -1 to 127 | Color of the points above the high limit line |
| Pn_CLimColor | Color | Pn_Plot-Color | -1 to 127 | Color of the points above the center line |
| Pn_LLimColor | Color | Pn_Plot-Color | -1 to 127 | Color of the points below the low limit line |
| Pn_ScaleMax | Float | 100 | float range | The max value of the y ruler |
| Pn_ScaleMin | Float | 0 | float range | The min value of the y ruler |
| Pn_ScaleFormat | | | | The format of the ruler marks |
| Pn_ScaleFont | Font | 1 | 1 to 71 | Font for the Y-ruler and unit for each plot. |

Table 44. XYPlot Properties

| Property | Type | Default | Range | Comment |
|------------------------------|-------|---------|-------------|----------------------------------|
| P _n _RulerValue | Float | None | float range | Cursor position along Y-axis |
| P _n _CurveVisible | Enum | True | True, False | Is this curve and labels visible |

(1) *n* is the curve number in the range [1 to 6].

The following sections provide guidelines for configuring XYPlot properties. Refer also to [Trend](#) on page 170 (XYPlot has some properties in common with Trend) and to [Basic Properties](#) on page 218.

Data Source

The data for each XYPlot curve is retrieved from two matrix elements - one for x-axis data, and one for y-axis data. Therefore, plotting all six curves would require data to be stored in 12 matrix elements. The input source is specified for each curve as the property P_n_Xserie and P_n_Yserie, where *n* is the curve number in the range [1 to 6]. The input source can be assigned a Matrix element column with a script. For instance:

```
For P1_Xserie - ##Matrix_1.Column[2] ;
For P1_Yserie - ##Matrix_2.Column[2] ;
```

Representation and Scatter Type

The P_n_Representation property determines how XYPlot curves are drawn. The options are:

- Scatter - the points are not connected.
- Curve - the points are connected.

The P_n_ScatterType property determines how the points of the plot curve are drawn. The options are: Circle, Square and Diamond. A point with ScatterType: Circle and ScatterStrength: 1 can be hidden beneath a curve with LineWidth: 1.

RulerAction

In Run mode and if the RulerVisible property is specified to True, a thread cross is shown in the plot area to show the x and y values of the current ruler position. The current ruler position is moved by clicking the mouse when the cursor is placed inside the plot area.

When the mouse is clicked, a new ruler position is calculated. The new X value is written into the property XRulerValue and the new Y values (for each visible curve) are written into the properties Pn_RulerValue.

Curve Data Color

If the Pn_LimitLineVisible property is set to True for a specific plot curve, three horizontal lines are displayed: a high limit line, a low limit line and a center line. The position of the high limit, low limit and center lines can be set by the properties Pn_LimitH, Pn_LimitL, and Pn_LimitC.

If the HLimColor and LLimColor properties are specified to a color other than the plot color, the points which exceed the high limit value are displayed in the HLimColor, and the plot points which are below the low limit value are displayed by the LLimColor. The function is illustrated in [Figure 122](#).

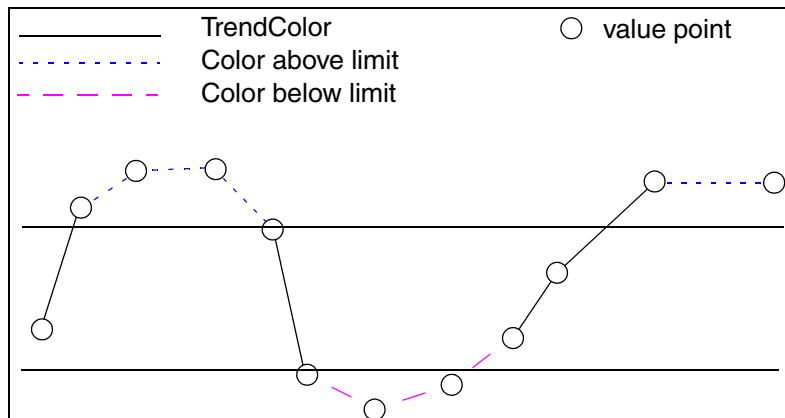


Figure 122. Plot Curve Drawn in Curve Mode

ActiveX Control

The toolbar button for the ActiveX display element is shown in [Figure 123](#).



Figure 123. ActiveX Button in Toolbar

The ActiveX display element lets you integrate pre-built display widgets into your display. For instance, you can add a calendar or QuickTime movie as an ActiveX display element, [Figure 124](#).

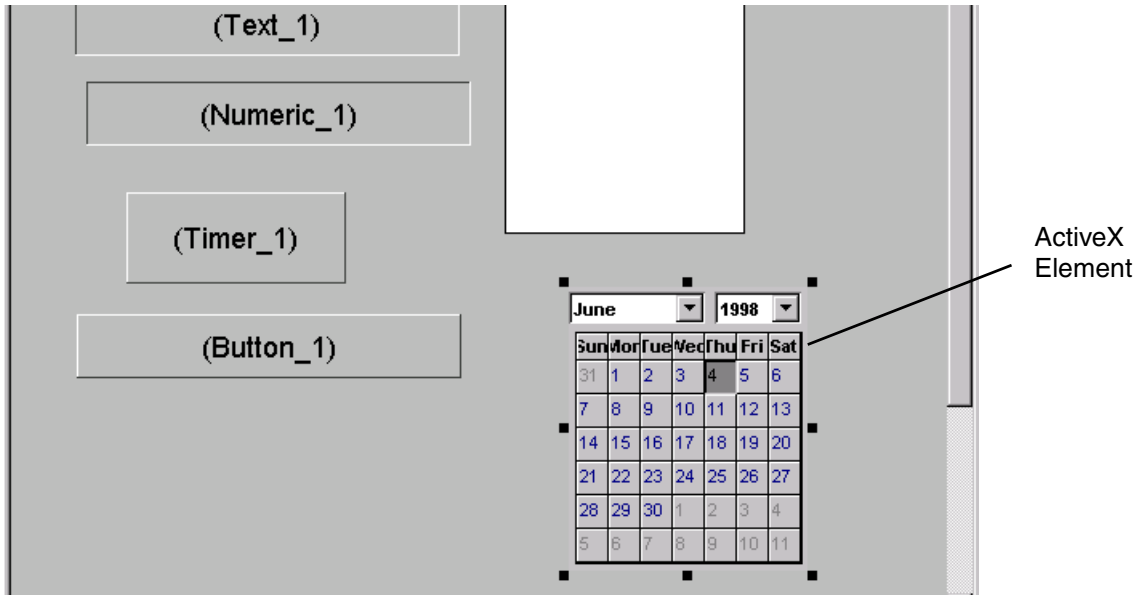


Figure 124. Adding a Calender as an ActiveX Control

Clicking the ActiveX toolbar button displays the Select ActiveX control dialog, [Figure 125](#).

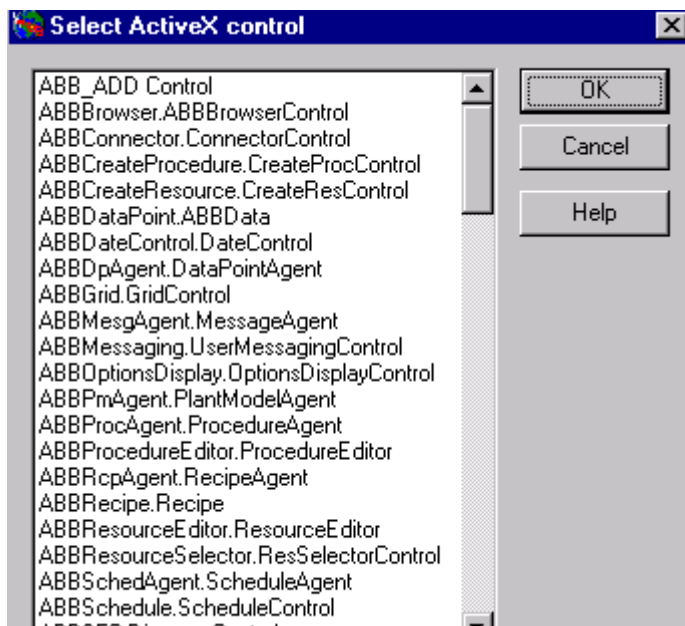


Figure 125. Select ActiveX Control Dialog

After selecting the control that you want to add, the procedure for adding the ActiveX display element is the same as for any other kind of display element. See [How to Insert a Display Element](#) on page 38 for details.

Configure [ActiveX properties](#) to set behavior and appearance characteristics.

The property list for the [ActiveX Control](#) display element is determined by the control you choose to add. All controls have a set of common properties as described in [Table 45](#).

Table 45. ActiveX Common Properties

| Property | Type | Default | Range | Comment |
|-------------------------|--------|-----------|------------|---|
| Name | String | Polygon_# | | Name must be unique within the display. |
| Comment | String | | 255 chars | Optional comment displayed in Run mode. |
| Visible | Enum | True | True/False | Whether or not line is visible in Run mode. |

Table 45. ActiveX Common Properties

| Property | Type | Default | Range | Comment |
|----------|---------|------------|-----------|--|
| X | Integer | As created | 0 to 1200 | X coordinate of polygon frame's upper left corner. |
| Y | Integer | As created | 0 to 1000 | Y coordinate of polygon frame's upper left corner. |
| Width | Integer | As created | 5 to 1200 | Width of polygon frame. |
| Height | Integer | As created | 5 to 1200 | Height of polygon frame. |

Basic Properties

For information on the following properties common to most display elements, see:

- X and Y coordinates, width, and height - [Positional Properties](#) on page 218
- [Name](#) on page 219
- [Comment](#) on page 220
- [Visible](#) on page 220
- [Foreground and Background](#) on page 221
- [Framewidth and Framestate](#) on page 221
- [DataQuery](#) on page 222
- [Action](#) on page 224
- [ReQuery](#) on page 225
- [DataChanged](#) on page 226

Positional Properties

The following properties determine a display element's position, [Figure 126](#):

X X axis coordinate for upper left corner of the display element's frame.

| | |
|--------|---|
| Y | Y axis coordinate for upper left corner of the display element's frame. |
| Width | Width of the display element's frame. |
| Height | Height of the display element's frame. |

These properties are common to all display element types, and to the display itself. They define the position and size of a display element. They can be modified in Build mode either by clicking and dragging, or via the [Properties Dialog](#). They can be modified in Run mode via display element actions defined using scripts. The geometry properties of the display itself can only be modified in Build mode. The X and Y coordinates 0,0 are located at the top left corner of the display.

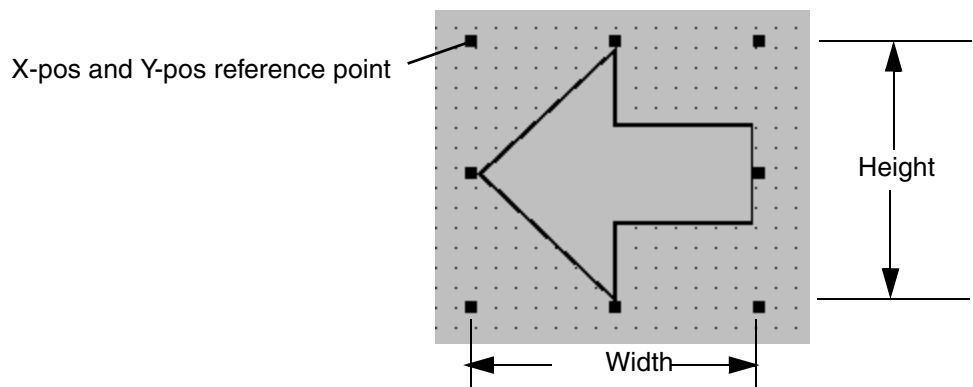


Figure 126. Geometry Properties

Name

Each display element instance must have a unique name. This name is used to reference the display element from another display element in the same display. The following default name is automatically assigned when a display element is inserted in the display via the toolbar: `<Type>_<Number>`.

Type is the type of display element, and *Number* is a unique number (not assigned to another display element of the same type). For example, the first display element of type Text is assigned the name: Text_1.

The display element name can only be changed in Build mode. The name of the display is always "Display". This is used when setting a property in the display.

Example: Changing the background color of the display

```
&Display.Background = 26;  
update("Display");
```

Comment

Each display element instance has a Comment property which may be defined on an optional basis. The property is a static text string of up to 255 characters. It is only active in Run mode. To display the Comment in the Run mode, click the right mouse button over the display element.

If a Comment is defined, a box containing the Comment text will appear for as long as the right mouse button is held down, as illustrated in [Figure 127](#). In this example, the Comment contains the tagname of the requested object.



Figure 127. Comment Property

The Comment can not contain a dataquery, but if desired you can modify the contents of Comment at runtime. The colors of the Comment can not be modified.

Visible

The Visible property lets you show or hide display elements on an individual basis, when the display is in Run mode. This property applies to all display elements, but not the display itself. It is interpreted as follows:

- | | |
|-------|-------------------------------------|
| True | The display element is visible. |
| False | The display element is not visible. |

Invisible display elements behave the same as visible ones, except they can not be seen. One exception is the Button. If you click on an invisible button no action is executed. However, you can still execute the action of an invisible button by a command from another display element. Another alternative is to set Visible

property of the button to true, and set the background color of the button to transparent (-1). This way, the button can not be seen, but the button action will be executed when the button area is clicked.

Foreground and Background

The Foreground and Background properties specify the pen (foreground) and fill (background) color of the display element, [Figure 128](#). Most display elements have both foreground and background properties; however, there are a few exceptions:

- The Display has no foreground.
- The Polyline has no background.
- The ABBButton has no background.

Text in Foreground Color



Background Color

Figure 128. Example, Foreground and Background

Framewidth and Framestate

Each element (except Display and Line) is surrounded by a Motif frame to show depth, [Figure 129](#). The Framewidth and Framestate properties control the appearance of the Motif frame. The state can be specified as *In* which makes the display element appear as pushed into the display, or *Out* which makes the display element appear as pushed out of the display.

Some display elements have no Motif-frames, some have two frames, and some have a Motif-frame depending on the definition of another property in the display element.

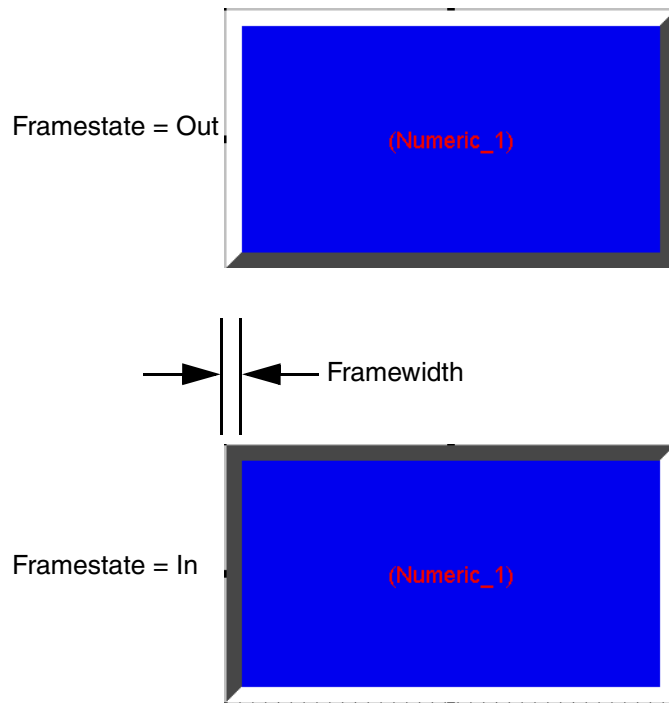


Figure 129. Example, Motif Frame

DataQuery

Some display element types have a DataQuery property for data retrieval. The query is specified by a script that executes either cyclically or on demand. The result is generally stored in the Value property (or an equivalent property). The Value property is read only, and is typically accessed by other display elements.

The query is executed initially upon invoking the display. Thereafter the query may be executed cyclically or on demand depending on how you configure the display element. When the object of a query changes, you must use the ReQuery action to issue a new query. An example of this is when the tag name in a DCS subscription query (dcssub) changes based on an item selected in a list. See [ReQuery](#) on page 225 for more information on requeries.

The DataQuery property may be defined as an SQL query, data statement, DCS subscription, or a constant value. Examples of these methods are provided below. Refer to [Section 6, Display Scripting](#) for details regarding DataQuery syntax.

Data Statement

The data statement is used in the DataQuery property of the matrix display element to query for data from OPC servers. An example application is illustrated in the [Section 3, A Quick Tutorial](#).

DCS Subscription

For DCS Subscription queries, whether the query is executed cyclically or on demand is determined by the access method part of the query. This can be defined as: dcs_demand, dcs_1s, dcs_3s, dcs_9s, or dcs_event.



Use only the approved cyclic rates.

This example gets the setpoint attribute for PID loop FC3014. The data is accessed via the CCF_PID_LOOP view in the Oracle database. The # symbol indicates the data type is float (real). The data request is executed cyclically, every three seconds:

```
#dcssub("FC3014","CCF_PID_LOOP","SETPOINT",dcs_3s)
```

This example gets the device command attribute for device HS3050. The data is accessed via the CCF_DEVICE_LOOP view in the Oracle database. The & symbol indicates the data type is integer. The data request is executed cyclically, every three seconds:

```
&dcssub("HS3050","CCF_DEVICE_LOOP","DEV_CMND",dcs_3s)
```

Constant Value

In this example, the DataQuery for a text display element is defined as a constant value and used as a label: "WATER TREATMENT"

SQL Query

For SQL queries, whether the query is executed cyclically or on demand is determined by the Update property. If you configure the Update property for execution on demand, you must use the ReQuery property to execute the query. This is described in [ReQuery](#) on page 225.

In this example, the query gets the value of the sysdate variable from the dual view:

```
&sql("select to_char(sysdate, 'hh24:mi') from dual")
```

In this example, the query gets the time and value from the numlogval view for a numeric log whose name is F_OIL,MEASURE:

```
&sql("select to_char(time, 'mm/dd/yy hh24:mi:ss'),  
entryvalue from numlogval where name = 'F_OIL,MEASURE'")
```



SQL queries are processed by the Oracle data provider. When the Oracle data provider is located on a System 800xA Information Manager, SQL queries can not be used to access data from the object types described in the *AdvaInform Object Types Reference Manual*. This includes basic objects such as AI, AO, DI, and DO, process objects such as CCF and TCL, system application objects such as NUMLOGVAL, and user objects built with the AdvaBuild Object Type Builder. Nor can SQLPlus be used to access these objects. These object types are accessible with SQL queries when the Oracle data provider is located on an HP-UX-based server.

When you need to access such objects from a System 800xA Information Manager, Inform IT - Open Data Access can provide access to these types of properties and values. Refer to *System 800xA Information Management Data Access and Reports manual (3BUF001094*)*.

Action

Some display element types have one or more Action properties. Actions are scripts that execute on certain events such as when a push button is activated or when data in a field changes.

In this example, the action of a button is configured to change the foreground color of Shape_94. The display will not reflect the change until the update statement is executed:


```
&Shape_94.Foreground=85;  
update(Shape_94);
```

You can configure actions to be nested within other actions, and inside other display elements. In this example, the action of a display element is configured to invoke the action of Button_1. You may want to invoke the action of a button this way when the button is made invisible.

```
execute("Button_1","Action");
```

Referencing Display Element Properties from other Display Elements

One common method for manipulating a display element property is to manipulate the property based on the value of another display element property. For instance, you can reference the Value property of a Numeric display element to change the color of a Shape display element:

```
if &Numeric_7.Value=0 then  
$Shape_2.Background="65";  
update(Shape_2);  
endif;
```

ReQuery

The ReQuery is an action that re-executes a query when the object to be queried has changed (for instance to re-query data for a List or a Combobox, or to change a DCS subscription in a DataQuery at runtime). This function is similar to the [Update](#) action.

Guidelines for Using Update and ReQuery

Update is used to get new data for a display element when the data source is internal (that is, the data resides within another display element in the display). For instance, you would use Update to get a new value for a numeric display element when the data source is a List display element.

ReQuery is used to get new data for a display element when you use a dcsub script or SQL query to get data from an external source where the object of the query can change.

The syntax for the ReQuery uses the script command ‘execute’. For example, to perform the ReQuery of a TextObject called MyText:

```
execute ("MyText" , "ReQuery" ) ;
```

The ReQuery action is applicable for all display element types that have a DataQuery property. These display element types are: Bar, Combobox, Edit, Gauge, List, MultiBar, MultiText, Numeric, PieText, Matrix, and XYPlot.



The one exception to the above rule is when the data source is a Matrix display element. In this case the query for new data must include a ReQuery action followed by an Update action. For example, if the data query for a numeric display element is: `$Matrix_1.Value[1,2]` ;, then the script to get new data for the numeric element would be:

```
execute ("Numeric_1" , "ReQuery" ) ;  
update (Numeric_1) ;
```

DataChanged

DataChanged is an action that is executed whenever the Value property changes. This can be used to signal other display elements to get updated values. The action is specified in a script.

A good example is provided in the demo display, Demomonitoring_1 (Plant Status). Open the properties window for the numeric_4 display element. This shows how the DataChanged property can be used to drive properties of other display elements. An excerpt from the script is shown in [Figure 130](#). Although this example uses a Numeric display element, the same principle holds true for other display elements.

```

if &Numeric_4.Value=0 then
&MultiBar_1.B1_DataQuery=12476;
update(MultiBar_1);
$Text_23.DataQuery="|0 1/min.|";
update(Text_23);
$Shape_18.Foreground="65";
$Shape_18.Background="65";
update(Shape_18);
$Text_5.DataQuery="|Filter stop|";
update(Text_5);
$Text_8.DataQuery="|-843 1/min.|";
update(Text_8);
$Text_24.DataQuery="|843 1/min.|";
update(Text_24);
$Text_56.DataQuery="|2473 1/min.|";
update(Text_56);
$Text_11.DataQuery="|Running|";
update(Text_11);
$Shape_19.Foreground="84";
$Shape_19.Background="84";
update(Shape_19);
$Text_57.DataQuery="|0 1/min.|";
update(Text_57);
$Text_60.DataQuery="|1846 1/min.|";
update(Text_60);
$Text_61.DataQuery="|0 1/min.|";
update(Text_61);
endif;

```

Figure 130. Example, DataChanged

Working with User Elements

A User Element is a collection of *Encapsulated member elements* which can be any combination of standard display elements, other user elements, or entire displays. The member elements visually appear on the display as if they are grouped, but are not represented in the [Element List](#) of the display. The member elements of a user element can not be accessed directly from outside the user element.

User defined properties (referred to as *user properties*) provide an interface between the user element and external objects such as other display elements and process objects. In build mode, User Properties appear as other properties in standard elements. It is recommended that you take a few minutes to work through the [User Element Tutorial](#) on page 232 to familiarize yourself with the methods for working with user elements.

There are two aspects to working with user elements: creating a user element definition, and inserting user element instances in displays. The basic procedure is as follows:

1. Add a new user element object to the Object Browser tree.

User elements are added as objects within a user type folder on the tree. This is equivalent to organizing displays in groups. Use the [Add Child](#) or [Add Sibling](#) commands in the [Object Browser](#) context menu to add a new user type folder.

For an example, see [Adding a New User Element Object](#) on page 234.

2. Configure the User Element definition.

This is similar to creating a new display. Insert the display elements that you want to include in the user element, and configure their respective properties. For an example, see [Inserting a Display Element in the User Element](#) on page 236.

The user element also has properties which are similar to display properties. These are described in [Properties for the User Element Definition](#) on page 249. To establish data connections with data sources outside the user element, you are required to configure user properties. This is done via the [User Properties Dialog](#).

For an example, see [Configuring User Properties](#) on page 236.

3. Instantiate the user element in one or more displays.

This is equivalent to inserting a standard display element in a display. When you create a user element instance in a display, the user element has another set of properties to configure, just like a standard display element. These are described in [Properties for User Element Instances](#) on page 259.

For an example, see [Creating a User Element Instance](#) on page 244.

4. You can protect user elements from unauthorized access. This is described in [Protecting Tool](#) on page 266.

For further information on the operation of instantiated user elements, see [Encapsulation](#) on page 229.

For guidelines on referencing user elements in scripts, see [Parent & This](#) on page 231.

Encapsulation

User elements facilitate display building by letting you save and re-use collections of display elements. When you cut and paste grouped display elements that are not user elements, a name mismatch will occur when one element references another element within the group. This is because each element is given a new name when pasted, but at the same time, all element references in the scripting code are maintained. This is illustrated in [Figure 131](#). When Group_1 is copied and pasted to create Group_2, the action for *Button_2* still references the *Text_1* display element. Consequently *Button_2* updates *Text_1* instead of *Text_2*.

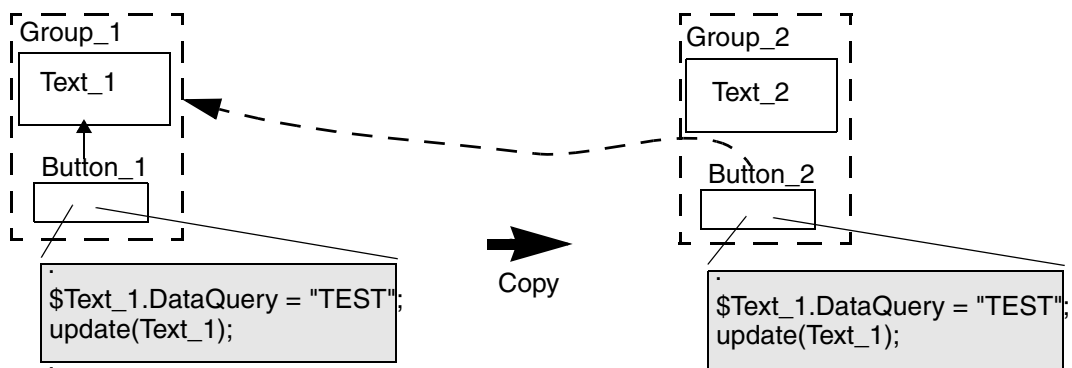


Figure 131. Copy/Paste Example, Not Using User Elements

In other words, each time a group of elements is copied, all element references must be modified to maintain the functionality.

When you build the display element group shown in [Figure 131](#) using user elements, you encapsulate the button and the text in a self-contained user element. The button and the text become member elements of the user element. They only

exist as elements within the scope of the user element. Seen from the outside, the user element is an entity. The button and the text display elements do not appear in the element list of the display.

Figure 132 shows the example from Figure 131 using user elements.

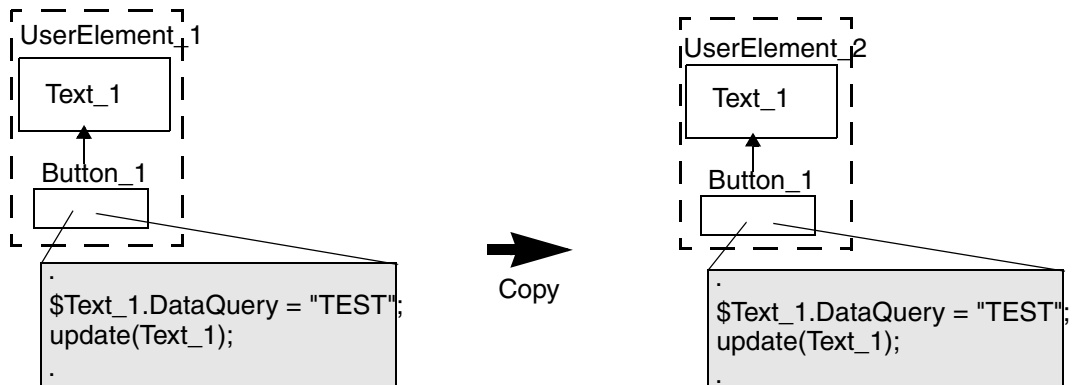


Figure 132. Copy/Paste Example With User Elements

A new name is assigned to the user element when it is copied; however, the member elements maintain the same naming for all instances of the user element. As all member element names and references are internal within the scope of the user element, there will not be any name conflicts nor mismatches.

When you insert a user element in a display, the user element operates just like any other standard display element.

The member elements inside a user element can be standard display elements as well as other user element instances.



DO NOT create an instance of a user element within the same user element. When you attempt to load such an element, the user element will try to load itself over and over again, thus entering an endless loop eventually causing a program crash.

It is recommended that you do not group member elements within a user element. Some elements may not be visible.

Member elements of a user element can only access other member elements within the user element itself. To interface the user element with other display elements and external (process) objects, you must use User Properties.

Parent & This

Parent and *This* are generic terms that you can use in scripts to replace literal references to user elements. This lets you copy and paste user elements and still maintain references from member elements to a property in the user element itself. Without the *Parent* and *This* terms, such references would not work properly. This is illustrated in [Figure 133](#) and [Figure 134](#).

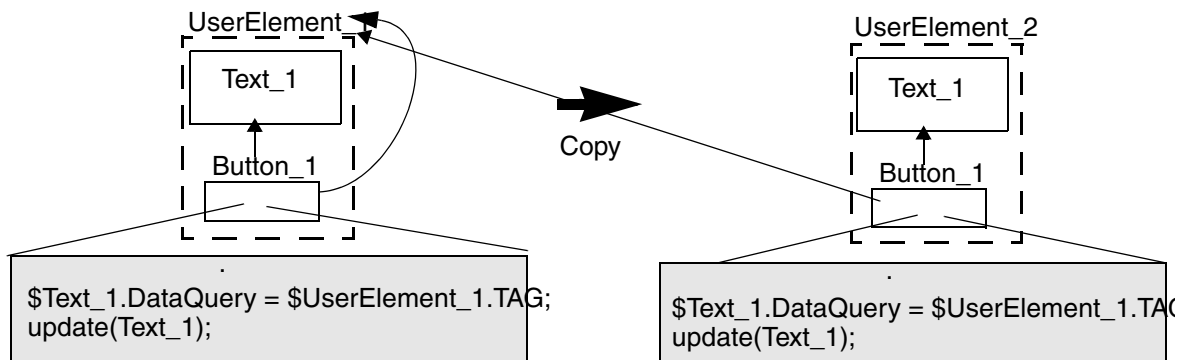


Figure 133. Example, References to User Element Without Parent/This Terms

When you copy and paste **UserElement_1** to create **UserElement_2**, the button in **UserElement_2** still tries to read from **UserElement_1** which is not the intention. In fact, because member elements of a user element can only reference within the user element itself, the button in **UserElement_2** cannot read anything directly from **UserElement_1**.

In [Figure 134](#) *Parent* and *This* replace the specific user element ID. The term *This* always references the element itself. The reference of *Parent* depends on the location of the element in which it is used:

- For an element in the display - *Parent* is the display
- For an element inside a *UserElement* - *Parent* is the *UserElement*
- Display has no parent

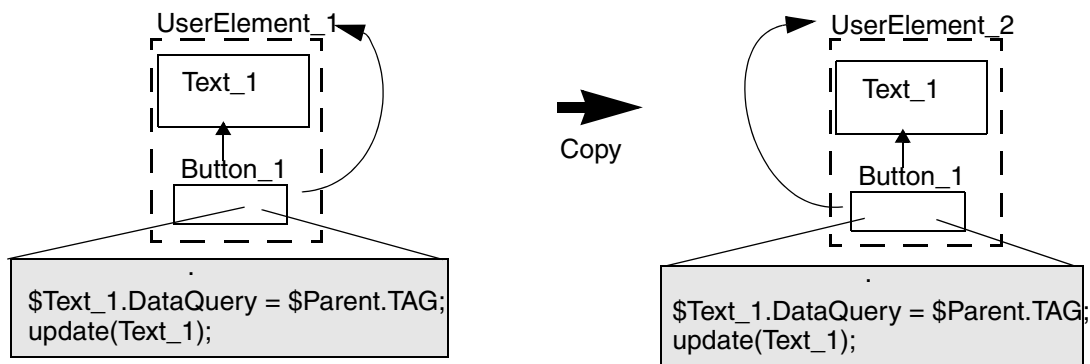


Figure 134. Example, References to User Element Using Parent Term

User Element Tutorial



In order to do the procedures described in this tutorial, you must already be proficient at basic configuration procedures such as adding a display element to a display and defining display element properties. These procedures are covered in [Section 3, A Quick Tutorial](#).

This tutorial takes you step-by-step through the process of building and using the following example user element. This user element will read the value of a process object that stores a floating point value such as an Analog Input (AI) object or a CCF loop. The user element consists of one standard Numeric display element (named AI_FIELD), which reads the floating point value.

The user element will require user properties as shown in [Figure 135](#).

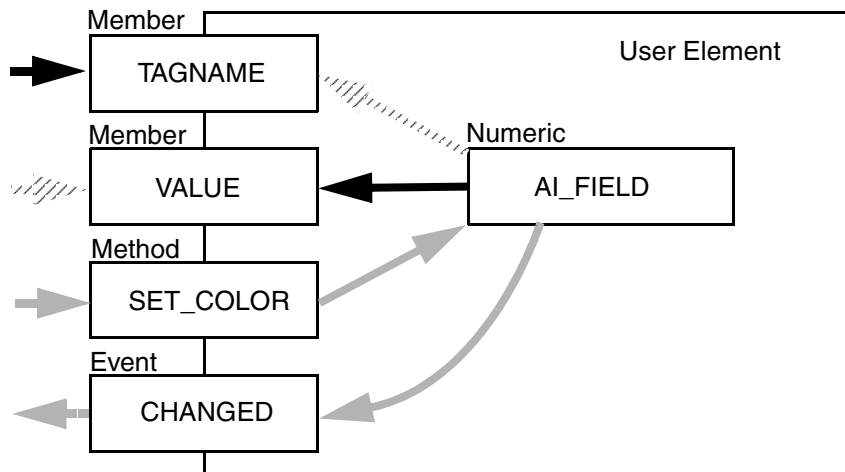


Figure 135. Example User Element

The function of these user properties are as follows:

- **TAGNAME** - Is a Member of type string. This property is used to specify the tagname of the process object that a value will be read from. The Numeric element (AI_FIELD) in the user element will reference the TAGNAME user property, and use the value entered there as the tagname in the data query for the numeric value.
- **VALUE** - Is a Member of type float. Each time the data query in the numeric element results in a new value, the new value is written to the VALUE user property. This user property can be read by display elements outside the user element.
- **SET_COLOR** - Is a Method with a parameter. The script for this method sets the background color of AI_FIELD to the color specified in the parameter.
- **CHANGED** - Is an Event which is called from AI_FIELD each time the value has changed. The script defined for the CHANGED user property is then executed.

Adding a New User Element Object

To create the new user element definition:

1. Start Display Services in build mode.
2. Create a new user type to store the user display element. The user type is a means of organizing new user element definitions into logical categories. This is equivalent to creating a group for storing displays.
 - a. Click on the display server icon in the object browser, and then right click and choose **Add child > USERTYPE** from the context menu, [Figure 136](#).



Figure 136. Adding a User Type

This displays the New Leaf dialog where you specify the name of the new user type.

- b. Enter **UE TEST**, then click **OK**, [Figure 137](#).

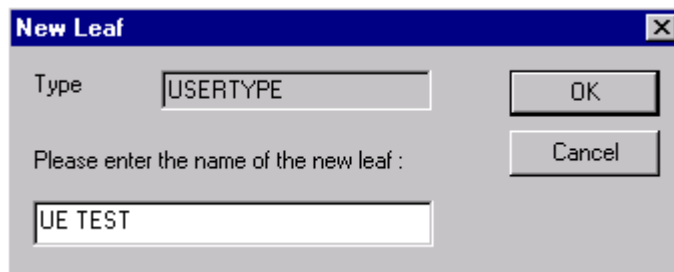


Figure 137. Specifying the User Type Name

This adds a new User Type Object to the object browser, [Figure 138](#).

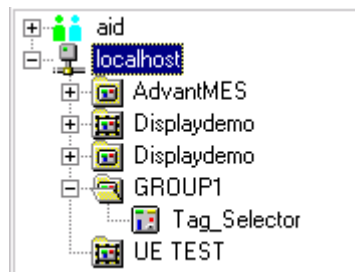


Figure 138. New User Type Added to the Object Browser

3. Create a new user element for the UE TEST user type. This is equivalent to adding a display to a group.
 - a. Click on the UE TEST object to select it, then right click and choose **Add child > USERELEMENT** from the context menu.
 - b. In the New Leaf dialog, specify AI TEST as the user element name, then click **OK**. This opens a new user element window, [Figure 139](#).

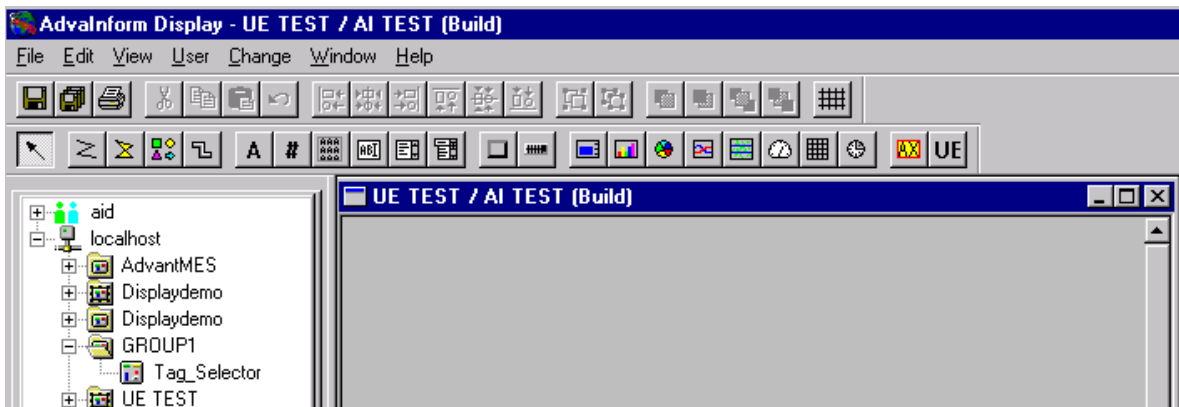


Figure 139. New Builder Window for AI TEST

Next, [insert a numeric display element in the AI TEST user element definition.](#)

Inserting a Display Element in the User Element

This procedure is similar to adding a display element to a standard display. To add a Numeric element to the user element:

1. Select the Numeric button and insert the Numeric element within the user element.
2. With the Numeric element selected, right click to open its the property dialog.
3. Select the Name property in the property list and change the name from the default (Numeric_1) to AI_FIELD, [Figure 140](#).

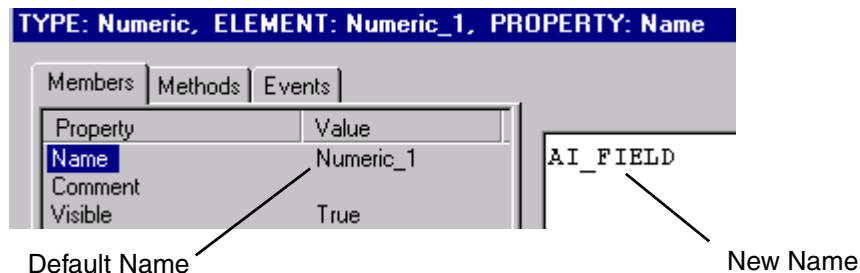


Figure 140. Changing the Name Property

4. Click **Apply** and then **Close**. The user element, AI TEST, now has a numeric element, AI_FIELD, as a member element encapsulated within it.

Next [configure the four user properties](#) (TAGNAME, VALUE, SET_COLOR, and CHANGED).

Configuring User Properties

1. Configure the TAGNAME user property for AI TEST. Remember that this user property will be used to specify the process object whose value will be read.
 - a. Display the Property dialog for the user element definition. To do this, first make sure the AI_FIELD element is NOT selected, then right click anywhere in the user element. The Property dialog for the user element is very similar to the Display Property dialog. The only difference is that some of the properties for user elements are not applicable for displays

and some of the properties for displays are not applicable for user elements.

- b. Select the **UserProperty** property. This displays the User Properties dialog, [Figure 141](#).

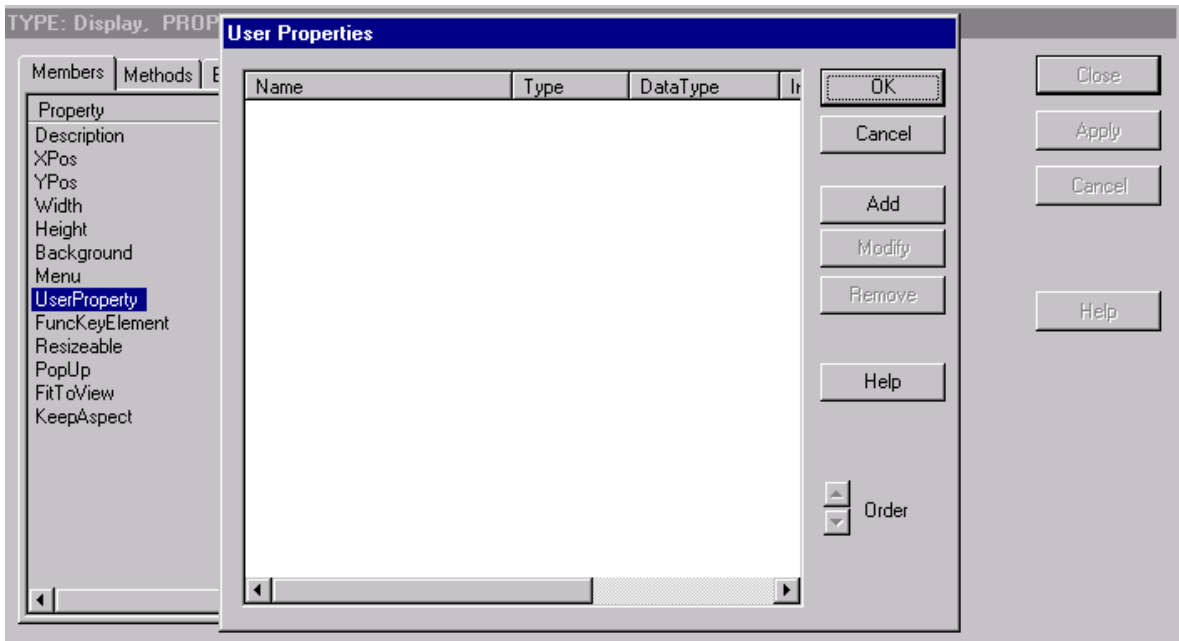


Figure 141. User Properties Dialog

- c. Click **Add**. This displays the Create New Property dialog, [Figure 142](#).
- d. Select, **Member** Property Type, **String** Data Type, and then enter **TAGNAME** in the Name field, [Figure 142](#).

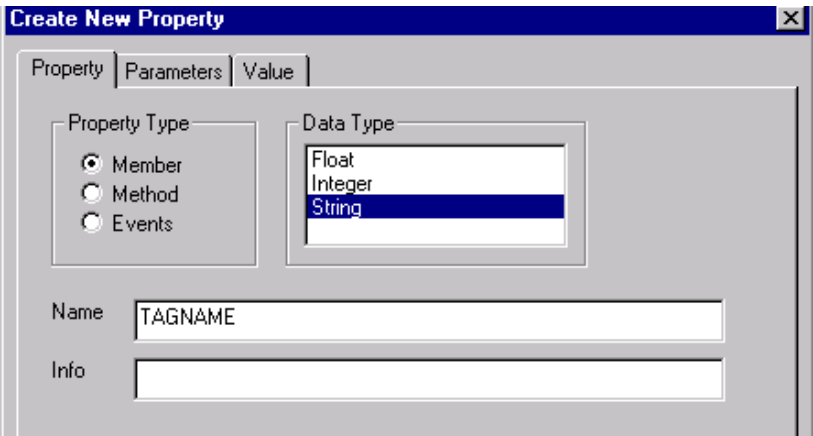


Figure 142. Create New Property Dialog

- e. Click **OK**. This closes the Create New Property dialog and returns focus to the User Properties dialog, with property definitions as you entered them, [Figure 143](#).

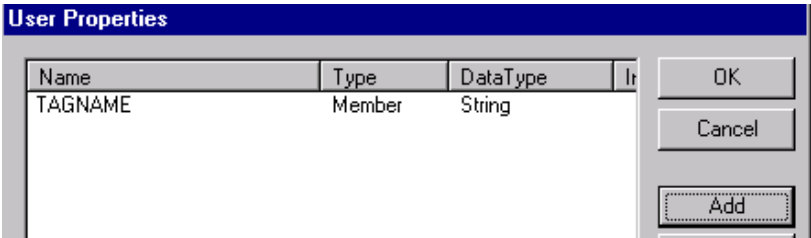


Figure 143. User Properties Defined

- 2. Repeat steps *c*, *d*, and *e* to add the VALUE (Member-Float), SET_COLOR (Method), and CHANGED (Event) properties to the AI TEST user element. Their respective functions are described in [User Element Tutorial](#) on page 232. The Create New Property dialog for each new property is shown in [Figure 144](#).

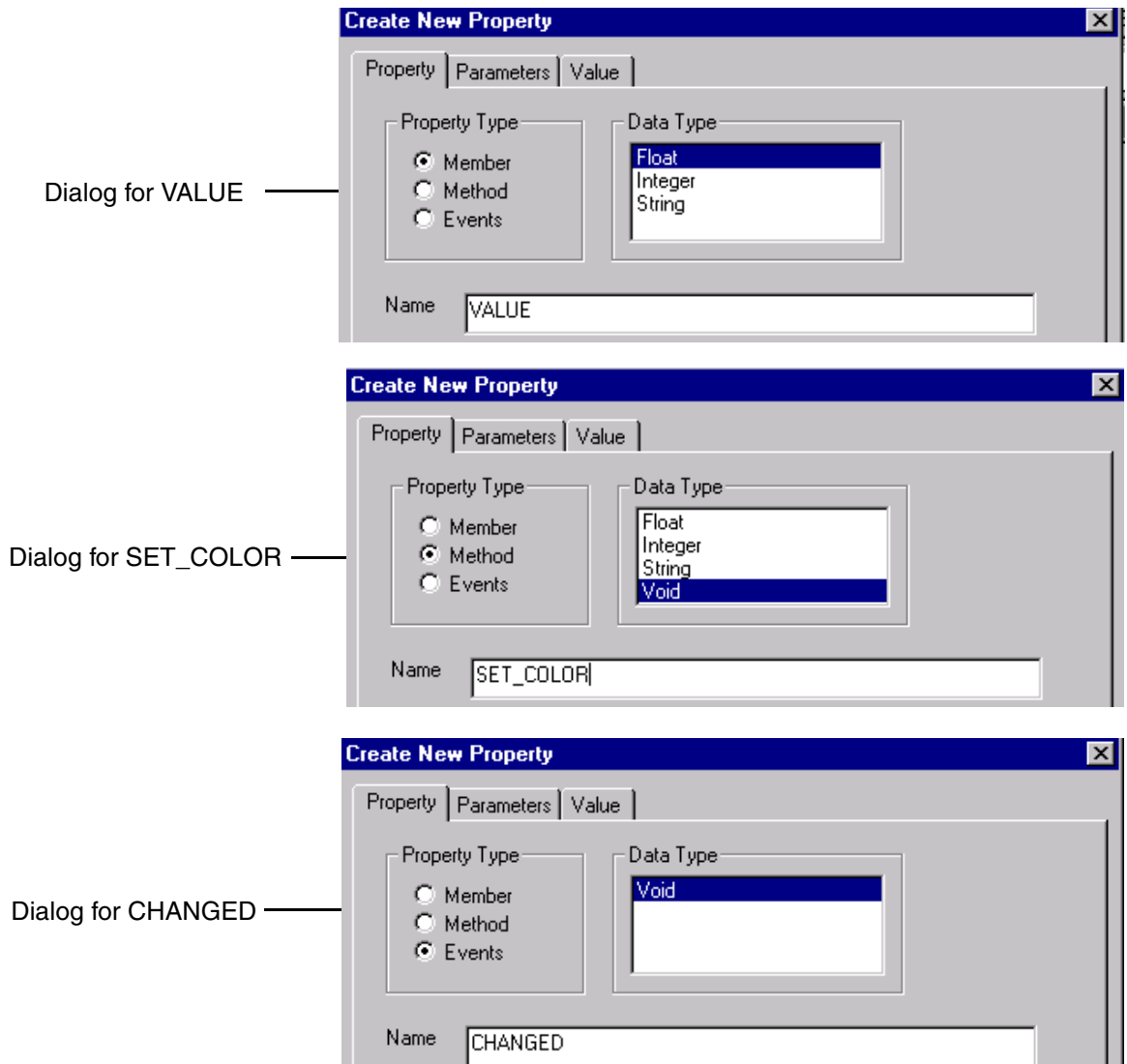


Figure 144. Dialogs for Adding VALUE, SET COLOR and CHANGED Properties

When you finish, the User Properties dialog will include all four properties, [Figure 145](#).

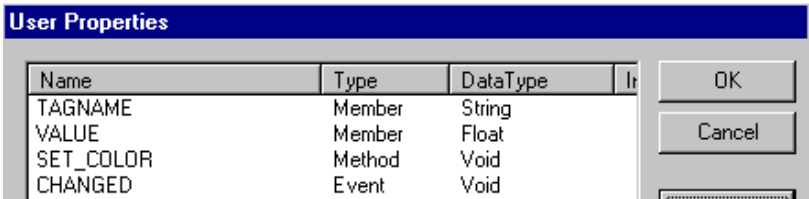


Figure 145. User Properties of AI TEST

Next, create a parameter and specify a value for the SET_COLOR property.

Modifying a User Property

SET_COLOR requires a script that sets the background color of the AI FIELD element according to a parameter value. To configure SET_COLOR:

- 1. Select SET_COLOR in the list and then click **Modify**. This displays the Modify Property dialog for SET_COLOR.
- 2. Click on the **Parameters** tab in the Modify Property dialog, Figure 146.

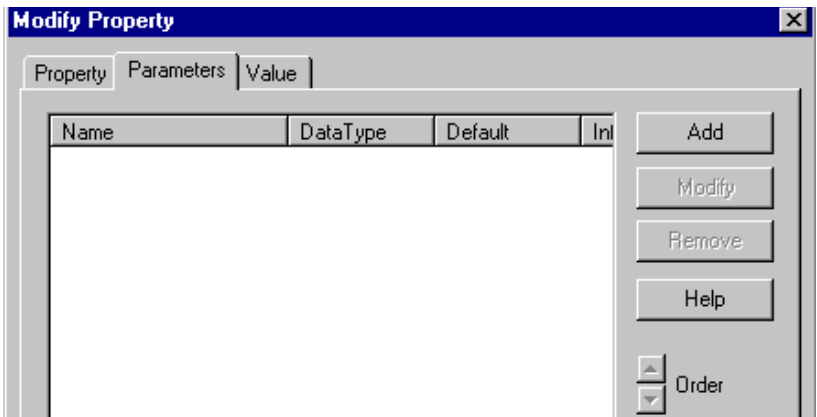
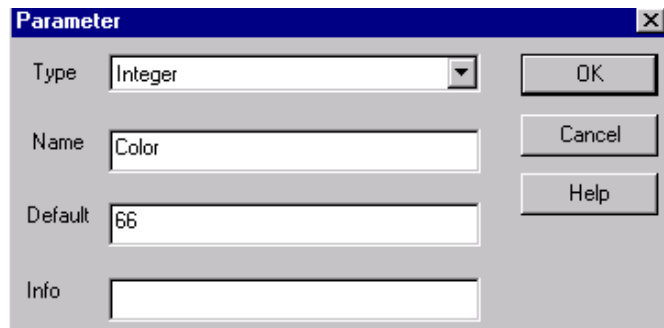


Figure 146. User Parameters Dialog for SET_COLOR

- 3. Click **Add**. This displays the Parameter dialog.

- Configure the new parameter as shown in [Figure 147](#). Select **Integer** as the parameter type, specify the parameter name as **Color**, and enter a default value of **66** (a shade of yellow).

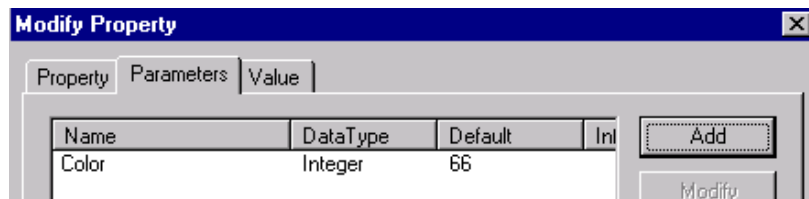


The **Parameter** dialog box contains the following fields and buttons:

- Type**: A dropdown menu with **Integer** selected.
- Name**: A text field containing **Color**.
- Default**: A text field containing **66**.
- Info**: An empty text field.
- Buttons**: **OK**, **Cancel**, and **Help** are located on the right side.

Figure 147. Parameter Definition Dialog

- Click **OK** to accept your entries. When keyboard focus is returned to the **Modify Property** dialog, the new parameter definition is displayed, [Figure 148](#).



The **Modify Property** dialog has three tabs: **Property**, **Parameters**, and **Value**. The **Parameters** tab is active, showing a table with the following data:

| Name | DataType | Default | Int |
|-------|----------|---------|-----|
| Color | Integer | 66 | |

Buttons **Add** and **Modify** are located to the right of the table.

Figure 148. New Parameter Definition

- Click on the **Value** tab. This displays a window where you can enter a script to update the background color of the user element.

This script sets the background color of the AI_FIELD numeric element to the value of the Color parameter defined above.

- Enter the script as shown in [Figure 149](#). When you are finished, click **Check** to check the syntax.

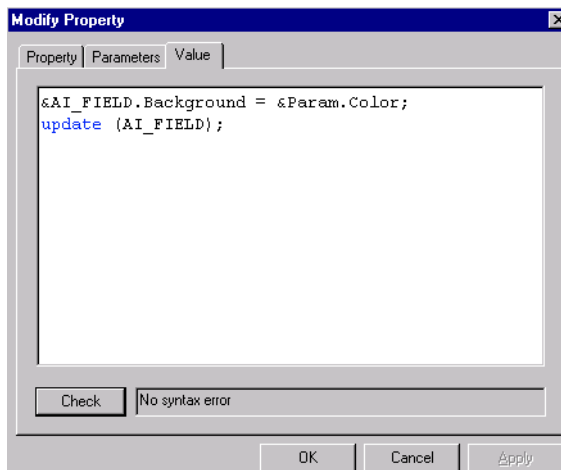


Figure 149. Script

8. Click **OK** to accept your entries if no errors are detected; otherwise, make the necessary corrections and try again.
9. When keyboard focus is returned to the User Properties dialog, click **OK**.
10. Close the Display Property window.
11. Choose **File > Save** from the main menu bar.

Next, [specify a data query for the numeric display element in AI_FIELD](#).

Referencing a User Property in a Script

The DataQuery property of the numeric display element in AI_FIELD must reference the TAGNAME user property:

1. Specify the DataQuery for the AI_FIELD numeric element.
2. Select the Numeric element AI_FIELD, right click, then choose **Properties** from the context menu. This displays the Property dialog for AI_FIELD.
3. Select the DataQuery property and enter the following script, [Figure 150](#):
`#dcssub ($Parent.TAGNAME, AI, VALUE, dcs_3s)`

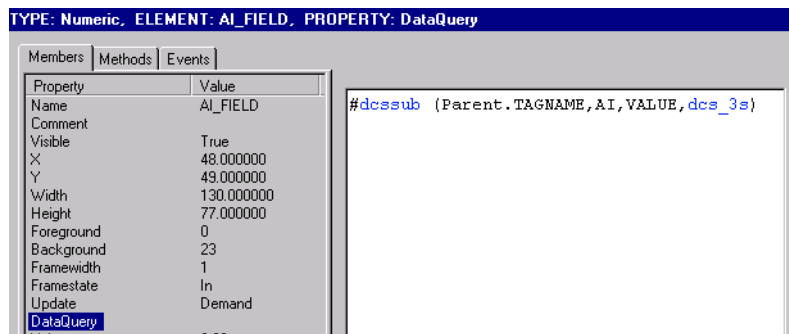


Figure 150. Specifying the DataQuery for AI_FIELD

This is a typical data query for a standard numeric element, except that the `Parent` term replaces the literal object ID specification. `Parent` is a generic reference to the user element which is the *parent* of this numeric member element (in this case `AI_TEST`). Thus, the user element definition can be instantiated without having to change the user element reference in the query. For more information on generic referencing, see [Parent & This](#) on page 231.

This query returns the value attribute of the AI object for the tag specified by `TAGNAME`. `TAGNAME` is defined on an individual basis for each user element instance ([Creating a User Element Instance](#) on page 244).

4. Click **Apply**.
5. Specify a DataChanged action for the numeric element.
 - a. Click on the **Events** tab in the Properties dialog to show the event-type property list.
 - b. Select the DataChanged property, and enter the following script, [Figure 151](#):


```
#Parent.VALUE = #This.Value;
Parent.CHANGED();
```
 - c. Click **Check** to check the syntax.

The first line of this script assigns the result of the data query for the numeric member element to the `VALUE` user property of the user element. Remember that properties of member elements within a user element are not accessible to

display elements outside the user element. By assigning the numeric member element value to the VALUE user property, the value is made accessible to external objects.

The second line of this script causes the action for the CHANGED user property to be executed. The CHANGED user property will be defined on an individual basis for each user element instance as described in [Creating a User Element Instance](#) on page 244.

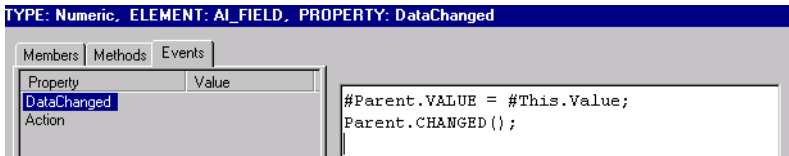


Figure 151. DataChanged Script

- 6. Click **Apply** and then **Close**.
- 7. Choose **File > Save**.

Next, [create an instance of AI TEST](#).

Creating a User Element Instance

Now add an instance of the new user element definition to a display:

- 1. Create a new display where you can insert the user element instance.
- 2. Click on the User Element button in the toolbar, [Figure 152](#).



Figure 152. User Element Button

This displays the Create User Element dialog.

- 3. Select UE TEST as user type, and AI TEST as the user element, [Figure 153](#).

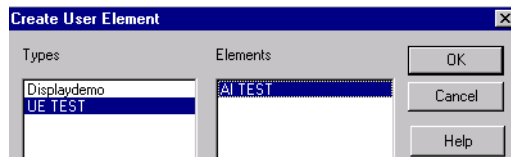


Figure 153. Create User Element

When you select an element and then move the cursor over the display area, the cursor changes to a cross hair so you can place the element in the display area. This procedure is equivalent to selecting a display element button from the toolbar. This puts an instance of the AI TEST user element in your new display, [Figure 154](#).

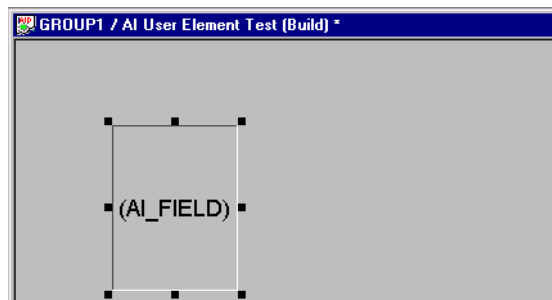


Figure 154. Instance of AI TEST User Element

The default name is User_1. The size of the instance should be the same as when you saved the user element definition, since the Resizable property has not been altered.

4. Display the Properties dialog for this user element, [Figure 155](#).

The new instance is assigned a default name User_1, just like a standard display element. You can change the name if you want to. The four user properties you created for the user element definition are available in the Properties List. TAGNAME and VALUE were created as member properties, so they are listed under the **Members** tab. SET COLOR is listed under the **Methods** tab, and CHANGED is listed under the **Events** tab.

Next, [test the new instance](#).

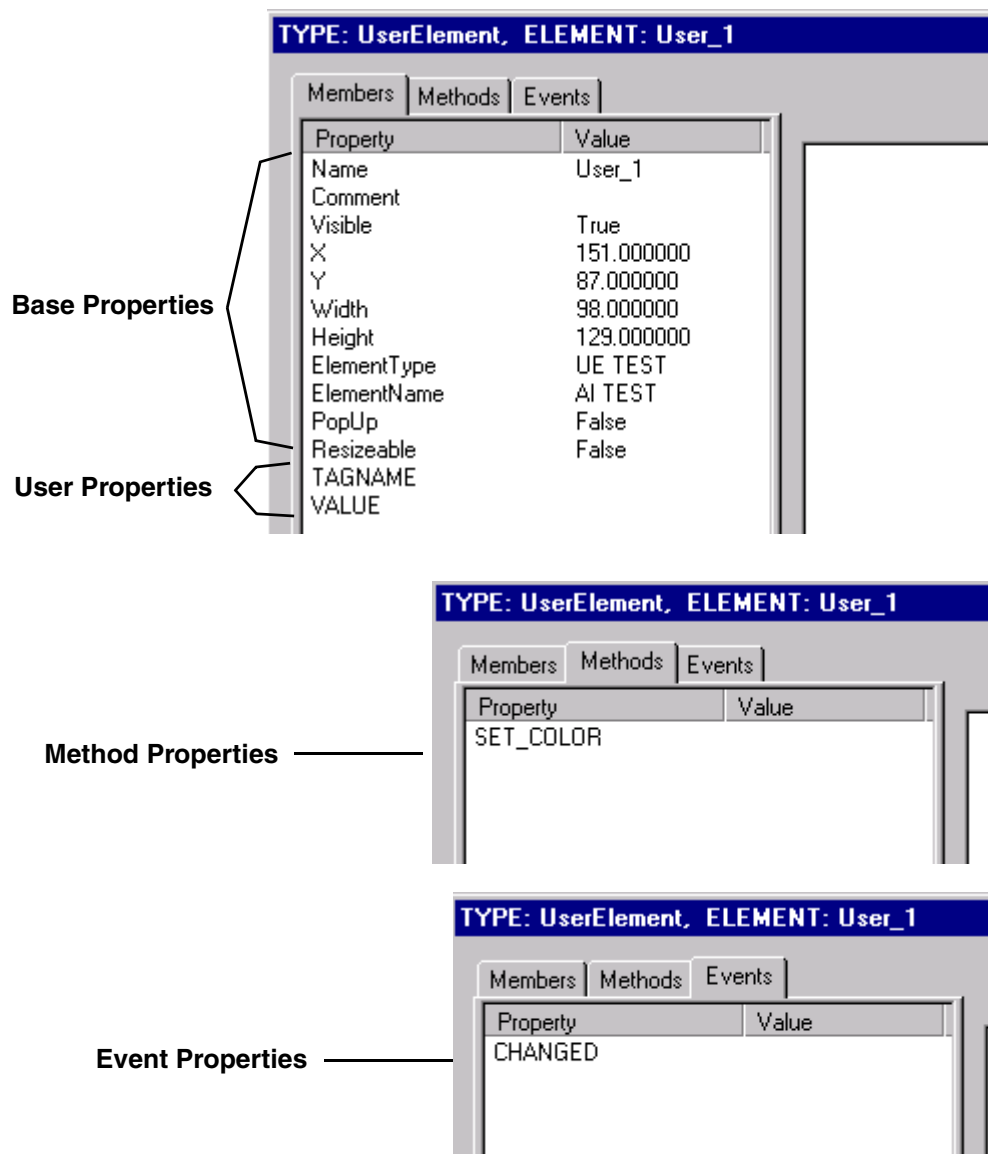


Figure 155. Property Dialog for User Element Instance

Testing the User Element

Verify that each of the user properties behaves as intended. First check if the AI_FIELD inside the user element can read the AI tagname entered in TAGNAME, and query data for it:

1. Select the TAGNAME property in the property list. Then, in the edit field, enter the tagname for an AI object in your system, [Figure 156](#).

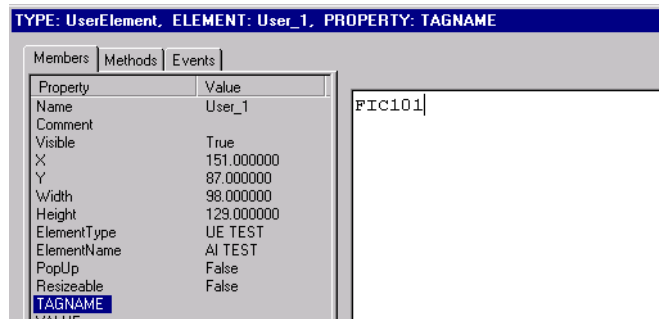


Figure 156. TAGNAME

2. Click **Apply** and then **Close**.
3. Choose **File > Save** from the main menu bar.
4. To run the display, select the display in the object browser, right click, and then choose **Run** from the context menu. When switching to run mode a green cross is displayed, [Figure 157](#). After a while the value of the AI object is displayed.

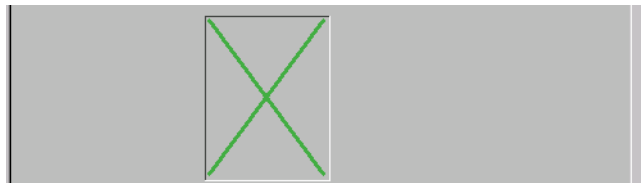


Figure 157. Green Cross in Run Mode

Next test the VALUE member of the user element:

1. Close the runtime display.
2. Add a numeric display element to the display with the user element.

3. Enter the following data query for the numeric element:
`#User_1.VALUE`
4. Click **Apply**.
5. Change the Update property to **Cyclic_1s**.
6. Click **Apply** and **Close**.
7. Choose **File > Save** from the main menu bar.
8. Run the display again.
9. Now the numeric field should display the same value as the user element. There may be up to 1 second delay.

To minimize the overall load of the display, you should not use cyclic reads but rather let the updates be event driven. In this case, you can use the CHANGED Event of the user element to notify the Numeric that VALUE has changed.

1. Close the runtime display.
2. Change the Update property for the numeric display element back to **Demand**.
3. Click **Apply**.
4. Open the property dialog for the user element.
5. Click on the **Events** tab, select the CHANGED property, and enter:
`update(Numeric_1);`
6. Click **Apply**.
7. Choose **File > Save** from the main menu bar.
8. Run the display again.

Now the user element will force the Numeric element to read VALUE, each time it has changed inside the user element, and you should see that the two values change simultaneously.

Test the SET_COLOR method by adding a toggle button which toggles the background color of the user element between red and green:

1. Choose **Action > Stop** from the main menu bar.
2. Add a Button display element to the display.

3. Enter the following script in the Action property for the Button display element:


```
if &This.Status then
    User_1.SET_COLOR(65);
else
    User_1.SET_COLOR(67);
endif;
```
4. Click **Apply**.
5. Specify the button type as **Toggle**.
6. Click **Apply** and **Close**.
7. Choose **File > Save** from the main menu bar.
8. Run the display again. Now the background color of the user element should toggle each time you activate the button.

Properties for the User Element Definition

Table 46 lists the properties for a user element.

Table 46. Display/User Element Properties

| Property | Type | Default | Range | Comment |
|--------------|---------|------------|---------------------|--|
| Description | String | NONE | up to 49 characters | Description of the display |
| Width | Integer | As created | 0 to 1200 | Width of the display. |
| Height | Integer | As created | 0 to 1000 | Height of the display. |
| Background | Integer | 16 | -1 to 127 | The display background Color. |
| UserProperty | String | NONE | | User Defined properties |
| Resizable | Enum | False | True/False | Determines if user element can be resized. |
| PopUp | Enum | False | True/False | Determines if user element is PopUp type. |

PopUp

PopUp determines whether the user element instance is a pop-up or not. PopUp type user element instances are displayed as a box with Motif frame in build mode, and are only visible in run mode when popped up. If the PopUp property is False, the user element instance will have the same appearance in both build and run mode.

Description

This property is only applicable for PopUp type user elements. The description is displayed in the title field of the pop-up dialog. There are three possible sources for the dialog title when the user element instance is popped up:

- **Comment** - If the Comment property has a value, it is used for the title.
- **Description** - If the Comment property has no value, but the Description property does have a value, it is used for the title.
- **Name** - If neither of the above applies, the Name is used for the title.

Width, Height

These properties are only applicable for PopUp type user elements. Width and Height define the size of the dialog when the user element instance is popped up. The size of non-PopUp user element instances is always the bounding rectangle.

Background

This property is only applicable for PopUp type user elements. It defines the background color of the dialog when the user element instance is popped up.

UserProperty

User properties are equivalent to properties in standard display elements. They are used to connect the user element to external objects such as other display elements and process objects. There are three types of user properties:

- **Members** - A member is a variable, which can be of type Integer, Float, or String. Members are comparable to public member variables in C++.

Members are used to transfer data to and from the user element. It is possible to write to and read from members both from inside the user element as well as from the outside.

- **Methods** - A method is a function which operates within the scope of the user element. Methods are comparable to public member functions in C++. A method can have parameters and return values.

Methods are called from outside the user element. You can supply parameters for the method along with the call. The user element processes the data, and may return a value to the caller outside.

- **Events** - An event is a function which operates outside the user element. Events have no analogy in C++. Like methods, events can have parameters and return values.

Events are typically used to notify an external object that some event has occurred (for example, a calculation is done, or a button has been activated). Events can return values to the caller inside the user element; however, this is not recommended. It makes the user element depend on external code, and more difficult to re-use.

Figure 158 shows the scope of the different user property types. The dashed line at the Method box indicates an optional return value.

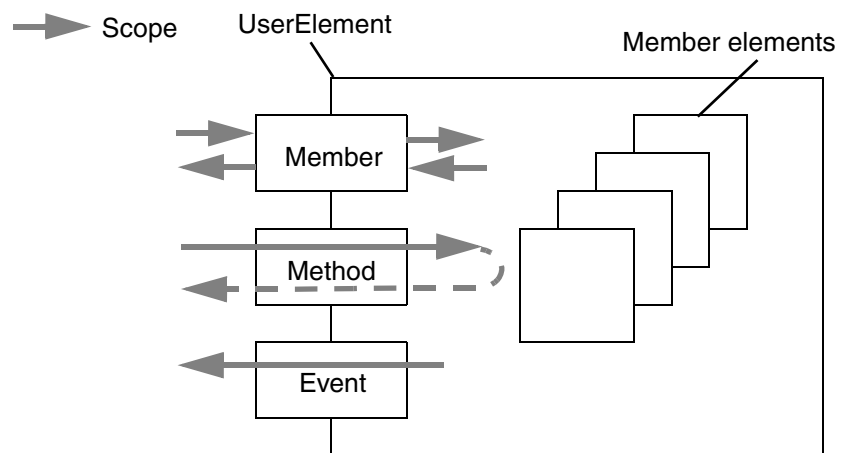


Figure 158. Scope of User Element Property Types

Selecting the UserProperty property displays the [User Properties Dialog](#). This dialog has various fields, buttons, and tabs for configuring user properties. For an example, see [Configuring User Properties](#) in the tutorial. For details regarding the User Properties dialog, see [User Properties Dialog](#) on page 252.

Resizable

This property determines whether the user element instance is resizable or not. If the PopUp property is True, Resizable will have no effect since the size of a popped up user element instance always is the size of the user element definition.

User Properties Dialog

User properties connect the user element to external objects such as other display elements and process objects. You create and manage user properties via the User Properties dialog, [Figure 159](#). To display this dialog for a user element:

- 1. Make sure all display elements within the user element are NOT SELECTED.
- 2. Right click in the user element and choose **Properties** from the context menu.
- 3. Select **UserProperty** in the Properties list.

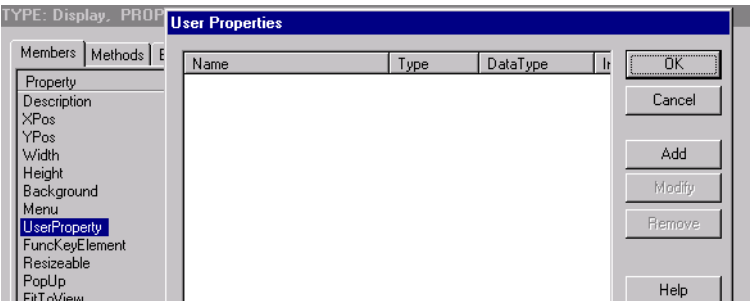


Figure 159. User Properties Dialog

This dialog is described in [Table 47](#). Use this dialog to:

- add a new user property via the [Create New Property Dialog](#). For an example, see [Configuring User Properties](#) on page 236 in the tutorial.
- modify an existing user property via the [Modify Property Dialog](#). For an example, see [Modifying a User Property](#) on page 240 in the tutorial.

- remove a user property. This is done via the **Remove** button.
- change the execution order of user properties via the **Order** button.

Table 47. User Properties Dialog

| Field/Button | Description |
|--------------|--|
| Name | This indicates the user property name. |
| Type | The type can be: Member, Method, or Event. |
| Data Type | Data type can be: Integer, Float or String. |
| Info | As an option, you can enter more descriptive information regarding this property. This information is only displayed in this dialog. |
| Add | This button lets you add a new user property. Clicking New displays the Create New Property Dialog , Figure 160 . |
| Modify | This button lets you modify an existing user property. This button is only active when a Property is selected in the User Properties dialog. Clicking Modify displays the Modify Property Dialog with the property definitions for the currently selected user property. You can either: <ul style="list-style-type: none">• edit the current property definitions under the Property tab• add parameters via the Parameters tab,• specify the value for this property via the Value tab |
| Remove | This button is only active when a property is selected in the User Properties dialog. It removes the selected property from the list in the User Properties dialog. |
| Order | The Up button moves the currently selected user property one position upwards in the list, and Down moves it downwards. |

Create New Property Dialog

The Create New Property dialog, [Figure 160](#), is used to add a new user property. To display this dialog, click **Add** in the [User Properties Dialog](#).

This dialog is described in [Table 48](#). To specify a value for the new property, or add parameters, select the property in the [User Properties Dialog](#), and then click **Modify**. This displays the [Modify Property Dialog](#).

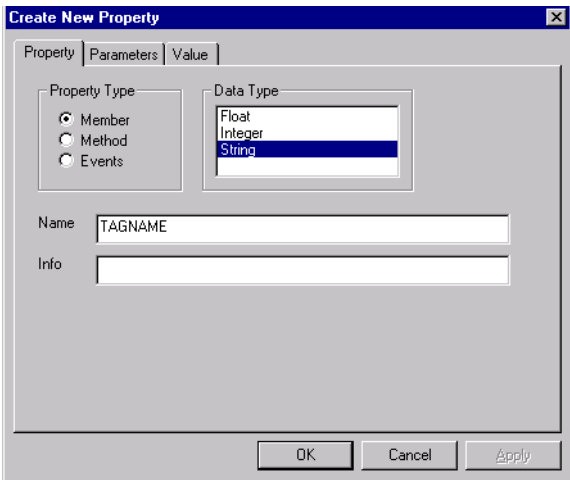


Figure 160. User Property Definition Dialog

Table 48. Create New Property Dialog

| Field/Button | Description |
|---------------|--|
| Property Type | Use the radio buttons to select the type of property that you want to add. The choices are: Member - Select this property type for variables or constants such as strings and integers. Method - Select this property type for functions that are called by the environment. Events - Select this property type for Actions (scripts) that execute on certain events, such as when a pushbutton is activated or a field changes. |
| Data Type | Use this window to select the data type for the user property. The choices in this window vary, depending on the Property Type selection. The choices are: Float, Integer, String, and Void. |
| Name | Enter the user property name in this field. The user property Name must be unique within the user element. You can not define a user property named 'X' because that is the name of one of the base properties. No spaces are allowed. |

Table 48. Create New Property Dialog

| Field/Button | Description |
|--------------|---|
| Info | You can enter a comment or miscellaneous information in this field. |
| Tabs | The tabs are only applicable when you modify an existing user property. |

Modify Property Dialog

To display the Modify property dialog, first select a property in the [User Properties Dialog](#), and then click **Modify**. The Modify Property dialog is used to:

- edit the current property definitions under the **Property** tab. The contents and operation of this tab are the same as for the [Create New Property Dialog](#).
- add parameters via the [Modify Property - Parameters Tab](#).
- specify the value for this property via the [Modify Property - Value Tab](#).

Modify Property - Parameters Tab

This tab is used to add parameters for Method and Event type user properties, [Figure 161](#). This tab is described in [Table 48](#).

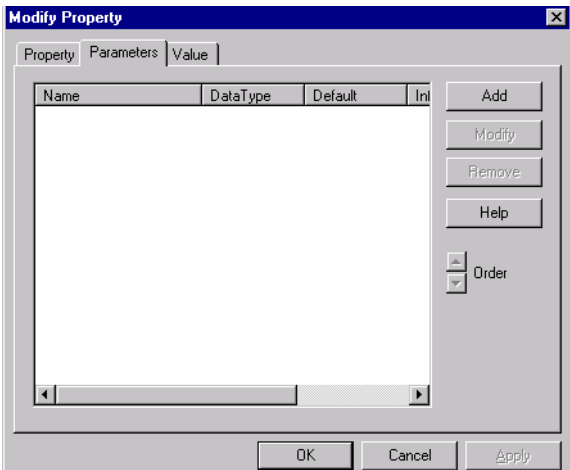


Figure 161. Modify Property Dialog

Table 49. Modify Property - Parameters Tab

| Field/Button | Description |
|--------------|--|
| Name | This field indicates the parameter name defined via the Parameter Dialog . |
| Data Type | This field indicates the data type specified via the Parameter Dialog . |
| Default | This field indicates the default value for the parameter specified via the Parameter Dialog . |
| Info | This field indicates any further description specified via the Parameter Dialog . |
| Add | This button lets you add a new parameter. Clicking New displays the Parameter Dialog , Figure 162 . |
| Modify | This button lets you modify an existing parameter. It is only active when a parameter is selected under the Parameters tab. Clicking Modify displays the Parameter Dialog with the parameter definitions for the currently selected parameter. You can edit these definitions as required. |

Table 49. Modify Property - Parameters Tab

| Field/Button | Description |
|--------------|---|
| Remove | This button is only active when a parameter is selected under the Parameters tab. Clicking Remove removes the selected parameter from the list. |
| Order | The Up button moves the currently selected parameter one position upwards in the list, and Down moves it downwards. The order of the Parameters is significant when calling the Method/Event. |

Parameter Dialog

The Parameter dialog, [Figure 162](#), is used to either configure a new parameter, or modify an existing parameter. To display this dialog:

- click **Add** in the **Parameters** tab of the [Modify Property Dialog](#).
- select an existing parameter to modify under the **Parameters** tab of the [Modify Property Dialog](#), and then click **Modify**.

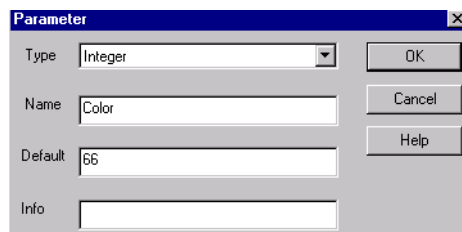


Figure 162. Parameter Dialog

Refer to [Table 50](#) for guidelines on using this dialog.

Table 50. Parameter Dialog

| Field/Button | Description |
|--------------|---|
| Type | You can enter the data type directly, or use the pull down menu. The choices are: Integer, Float or String. |
| Name | Enter the parameter name in this field. |

Table 50. Parameter Dialog

| Field/Button | Description |
|--------------|--|
| Default | Enter a default value in this field according to the selected Type . |
| Info | As an option, you can enter more descriptive information in this field. |

Modify Property - Value Tab

The **Value** tab lets you specify a value for member and Method type properties, [Figure 163](#). Events do not have a value.

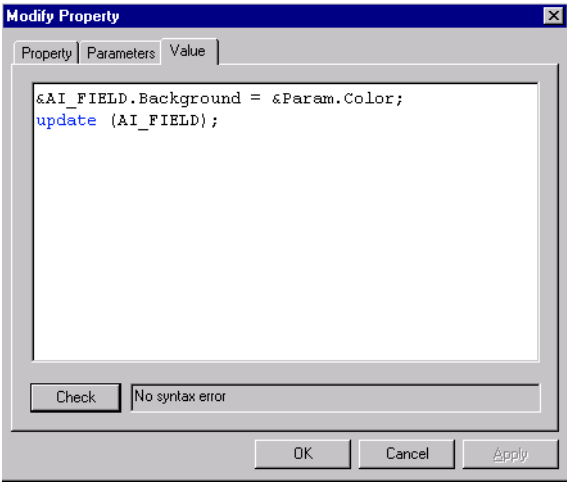


Figure 163. Modify Properties - Value Tab

Enter the value directly in the edit window. For member type properties, enter a Float, Integer, or string according to the data type.

For method type properties, enter a script. Use the **Check** button to check the script syntax.



A matrix element can not be passed into a user element as a parameter.

Properties for User Element Instances

Setting properties in a user element instance does not affect any other instances. Settings in the basic user element definition can not be overruled from a user element instance.

[Table 51](#) below lists all the properties for an instance of a user element as they appear in the Property dialog. The Apply To column indicates whether a property applies when the user element is in normal mode (N), in [PopUp](#) mode (P) or both. The shaded rows indicate Read Only properties which are set in the user element definition and not are changeable in the instance.

The user element instance has two actions which can be called by the user of the user element. Both actions are only applicable for user elements in [PopUp](#) mode. See [User Element Actions](#).

Table 51. Base User Element Instance Properties

| Property | Type | Default | Range | Comment | N or P ⁽¹⁾ |
|-----------------------------|---------|-----------------|------------|---|-----------------------|
| Name | STRING | User_# | | Name must be unique | N/P |
| Comment | STRING | | 255 chars | Optional comment | N/P |
| Visible | ENUM | True | True/False | Visibility in run mode | N |
| X | INTEGER | As created | 0 to 1200 | X Coordinate of upper left corner of instance | N |
| Y | INTEGER | As created | 0 to 1200 | Y Coordinate of upper left corner of instance | N |
| Width | INTEGER | As created | 0 to 1200 | Width of instance | N |
| Height | INTEGER | As created | 0 to 1200 | Height of instance | N |
| ElementType | STRING | Type name | | The type of the user element definition | N/P |
| ElementName | STRING | Definition name | | The name of the user element definition | N/P |
| Resizable | ENUM | False | True/False | Determines whether the instance should be resizable | N |

Table 51. Base User Element Instance Properties

| Property | Type | Default | Range | Comment | N or P ⁽¹⁾ |
|----------|---------|---------|-----------------------|--|-----------------------|
| PopUp | ENUM | False | True/False | Determines whether the instance should be in Normal or PopUp mode | N/P |
| PopUpX | INTEGER | | Any screen coordinate | Determines the location where the instance will pop up. PC-Client only | P |
| PopUpY | INTEGER | | Any screen coordinate | Determines the location where the instance will pop up. PC-Client only | P |
| Modal | ENUM | True | True/False | Determines whether popped up instance is modal or not | P |

(1) N = Normal Mode, P = PopUp Mode

Name

This is the name of the user element instance. By default instances are named “User” with a sequential number to distinguish each instance (User_1, 'User_2, and so on). The name can be changed, but it must be unique within the display, as with any other display elements.

Comment

The function of the comment property depends on whether the user element instance is in PopUp mode or not.

Comment in Normal mode

In normal mode, the comment is text displayed when you right-click on the user element instance. The individual elements inside a user element can have their own comment. Thus a user element instance can have different comments over an area.

When more than one comment is defined for a user element, which comment is displayed for the element is determined by the order of the elements, and by the following rules:

- If the cursor is over one or more elements, show the comment of the first element that has a comment defined for it.
- If the cursor is not over one or more elements, or if no elements have comments, show the comment of the user element instance.

This is illustrated in [Figure 164](#). The left side shows how comments have been defined for some member elements within a user element. The user element instance is assigned the comment "The Instance" via the property list. The right side shows which comment is displayed when you click on a specific area of the user element instance.

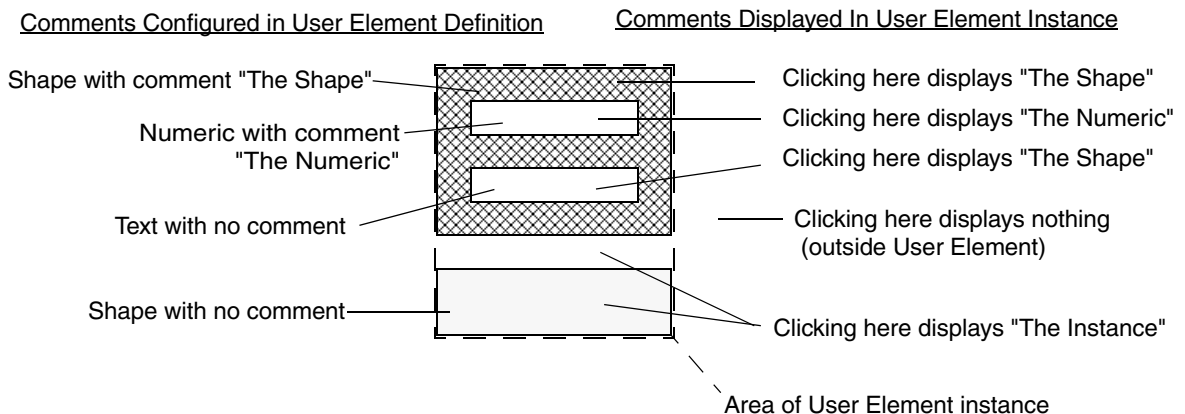


Figure 164. Elements with Comments within a User Element

Comment in PopUp mode

When the user element instance is in PopUp mode, the comment can be used to set the dialog title when the user element instance is popped up:

- **Comment** - If the Comment property of the user element instance has a value, that value is used for the title.
- **Description** - If the instance does not have a Comment, but the Description property of the user element definition has a value, that value is used for the title.
- **Name** - If neither Comment nor Description have a value, the name of the user element instance is used for the title.

When popped up, the comments of the elements inside the user element can be displayed via the right mouse button. The comment of the user element itself is not displayed this way.

Visible

The Visible property is only applicable when the user element is of type Normal and not PopUp. It is used to toggle the visibility of a user element, just like any other element. When set to False the user element is visible in build mode, but not in run mode.

When the user element is invisible in run mode, it can be accessed via scripting code, just as if it was visible.



If a user element contains Edit, Combobox or List elements, those elements are NOT made invisible by making the user element invisible. If this functionality is required, the user element must have methods to hide or show the member elements.

X, Y, Width, Height

The function of these properties depends on the mode of the user element.

Position and size in Normal mode

If the Resizable property is False, the X and Y properties are changeable, the Width and the Height are constant. The instance can be placed anywhere on the display, but not resized.

If the Resizable property is True, all four properties are changeable. The instance can be moved and resized

Position and size in PopUp mode

When the user element is of type PopUp, the Resizable property has no effect. In build mode, the instance of the user element is shown as a motif frame which you can position and resize as you wish. In run mode the instance is invisible until popped up, and the size when popped up is the size the definition was originally saved at, disregarding the size of the instance in build mode.

ElementType & ElementName

These properties specify the type and name of the user element definition. They can not be modified. They are intended to help you find the source of a user element. The location is:

<AID_DATA_DIR>/<ElementType>.uet/<ElementName>.ue<dlb>

where:

- AID_DATA_DIR - is specified in the AIDisplayenvironment file (for HP-UX 9.05 platforms) or environment file (for HP-UX 10.20 platforms) in the `etc` directory of the AdvanInform Display installation directory.
- ElementType is the ElementType property. It indicates a directory with the name of the element type with extension `uet`.
- ElementName is the ElementName property. It indicates a file with the name of the element, with extension `ued` or `ueb`.
- `ued` - is a standard definition file. It is a standard readable ASCII file.
- `ueb` - is a binary definition file. The file has been crypted and is not readable, unless by a user with the correct password or by Display Services. For more information on the crypting tool, see [Protecting Tool](#) on page 266.

The following are examples of directory and file specifications for HP-UX 9.05:

```
/products/data/Display/UE_TEST.uet/AI_TEST.ued  
/products/data/Display/MY_TYPE.uet/MY_ELEMENT.ueb
```

The equivalent directory and file specifications for HP-UX 10.20 are:

```
/home/opt/advant/Display/UE_TEST.uet/AI_TEST.ued  
/home/opt/advant/Display/MY_TYPE.uet/MY_ELEMENT.ueb
```

Resizableable

The Resizableable property is set in the user element definition, and it can not be changed in the user element instance. It is only applicable for user elements in Normal mode, not in PopUp mode. See X,Y, Width, Height for more information regarding the behavior of the user element based on the Resizableable property.

PopUp

The PopUp property is set in the user element definition, and it can not be changed in the user element instance. This property determines the mode of the user element:

- False - mode is Normal. The user element is visible in the Build mode.
- True - mode is PopUp. In build mode user element appears as a 3D-frame. In run mode it is invisible, until popped up in a dialog. When popped down, the user element is invisible.

PopUpX & PopUpY

When popped up the user element is by default placed at the center of its parent dialog. In the PC-client, it is possible to specify the coordinates where the user element should pop up. If popped up and moved, the last position of the user element is written to these properties when the user element is popped down again. This means that the next time the user element is popped up, it will be displayed at the same location as before.

Modal

Modal applies to user elements in PopUp mode only. The setting (True/False) determines the behavior of the dialog in which the user element is popped up:

- True - All input focus such as select with the mouse or entering data from the keyboard is only possible in this dialog, for as long as it is popped up. If this dialog pops up a new modal dialog, the input focus goes to the new dialog.
- False - Input focus is not locked by this dialog. You can select both in this dialog, the display or other dialogs.

[Figure 165](#) shows four examples of the Modal property. These examples are described below:

- Example 1 - user element 'a' is popped up. The Modal property of 'a' is True so only 'a' has input focus, and the display is inactive, indicated by dimming display '1'.
- Example 2 - user element 'a' is popped up. The Modal property of 'a' is False, so both 'a' and display '2' have input focus.

- Example 3 - user element 'a' is popped up and Modal property is True. This is the same situation as in Example 1. user element 'b' is popped up from 'a' and the Modal property of 'b' is also True. This means that 'b' now takes the input focus from 'a' indicated by dimming 'a'. When 'b' is popped down 'a' regains the input focus.
- Example 4 - user element 'a' is popped up and Modal property is False. Both 'a' and display '2' can have input focus (the same as Example 2). user element 'b' is also popped up from the display, with Modal property set to true. 'b' now takes the input focus from both the display and 'a'. When 'b' is popped down, the situation is as in Example 2 again.



Due to differences in the platforms of the display clients, the implementation of pop up dialogs vary slightly from X to the PC client. In order to secure proper functionality, it is strongly recommended that multi-level dialogs (when one dialog is popped up from another dialog) are implemented using only modal ('Modal' = True) user elements.

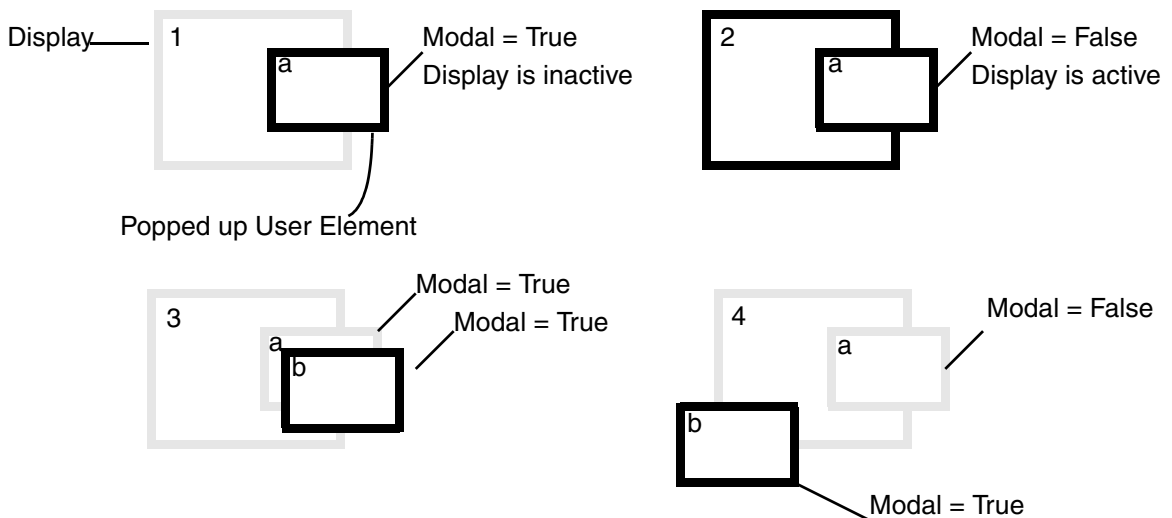


Figure 165. Examples of Modal Property

User Element Actions

The user element instance has two actions which can be called by the user of the user element. Both actions are only applicable for user elements in PopUp mode.

PopUp. The PopUp action is used in run mode to pop up a user element. The syntax is:

```
execute(<User Element instance name>,PopUp);
```

The PopUp action can be called from the Action of another element, for example a PushButton:

```
execute(User_1,PopUp);
```



Due to differences in display client platforms, the implementation of pop up dialogs vary slightly from X to the PC client. In order to secure proper functionality, it is strongly recommended that the call to 'execute(...,PopUp)' is the last scripting code to be executed in the Action.

PopDown. The PopDown action is used in run mode to pop down a user element. The syntax is:

```
execute(<User Element instance name>,PopDown);
```

If the user element is not Modal, the PopDown action can be called from the Action of another element. for example a PushButton:

```
execute(User_1,PopDown);
```

If the user element is Modal, the PopDown action can be called from the Action of an element inside the user element, for example a PushButton:

```
execute(Parent,PopDown);
```

Protecting Tool

The protecting tool lets you protect user element definitions from unauthorized modifications.

When you protect a user element definition file, you provide a password for a crypting algorithm, and a new (crypted) file with the extension 'ueb' (user element

Binary) is created. The binary file is not readable using a text editor, but the display server can read the file using a master password.

If you wish to revert the protected file back to an ordinary user element definition file, this can be done by unprotecting the binary file using the same password as when the file was protected.

Crypting is typically done when you have finished developing and testing user elements, and are ready to 'lock' the project. The process could be:

- Design, create and test user element definitions
- Protect all definition files by converting them to binary files
- Remove all the (readable) definition files

To protect the files, *UEprotect.exe* must be launched from **C:\Program Files\ABB Industrial IT\Inform IT\Display Services\Server\Utils\UEprotect.exe**. The GUI is launched as shown in [Figure 166](#).

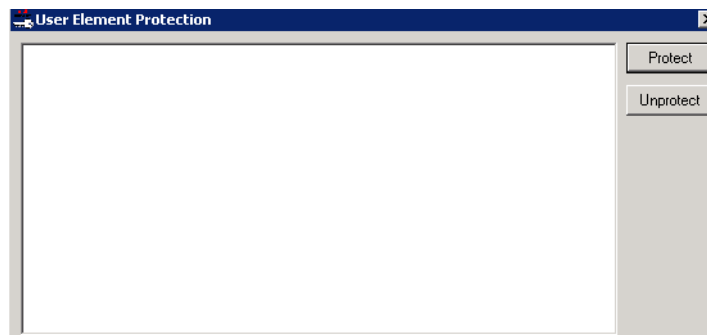


Figure 166. User Element Protection



If both, a protected and an unprotected file exists in the same directory, the contents of the protected file is used for loading.

Select the file to protect or unprotect and use the buttons to perform the desired action.

Trace Log

The Trace log displays system error messages and user defined messages with time stamps, [Figure 167](#). Messages are sent to the trace log via the trace script. To open the trace window, choose **Help > Diagnostics > Trace Log** from the main menu bar.

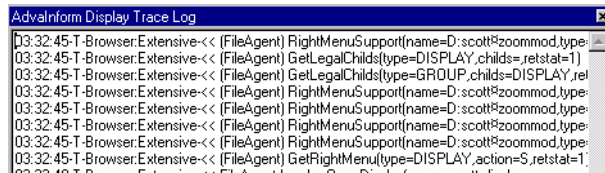


Figure 167. Example Trace Log

You can selectively enable and disable trace variables to control the amount of information displayed in this window. To do this, right-click inside the Trace Log window, then choose **Settings...** from the context menu. This displays the Trace Settings dialog, [Figure 168](#).

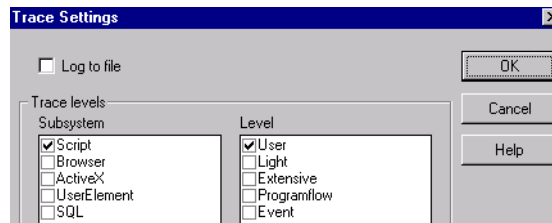


Figure 168. Trace Settings Dialog

Check the variables that you want to enable, and uncheck the ones you want to disable. The **Log to File** check box lets you log the trace messages to a file. The file is specified when you configure the PC client environment as described in [Setting Up PC Client Environments](#) on page 21.

The default is:

C:\Program Files\ABB Industrial IT\Inform IT\Display Services\Client\Log

You can clear the Trace Log window, or copy the contents to the paste buffer via the context menu.

Communication Statistics

This dialog displays communication statistics, [Figure 169](#). To open this dialog, choose **Help > Diagnostics > Communication Statistics** from the main menu bar.

| Synchronous | | Asynchronous | |
|-----------------|-----------------|-----------------|-----------------|
| RX | TX | RX | TX |
| Bytes: 92394 | Bytes: 5718 | Bytes: 0 | Bytes: 4402 |
| Messages: 25 | Messages: 245 | Messages: 0 | Messages: 220 |
| Msg. pending: 0 | Msg. pending: 0 | Msg. pending: 0 | Msg. pending: 0 |

Figure 169. Communication Statistics

Synchronous includes communications such as display load and sql() script command. Asynchronous includes such communications as data received in a matrix element and dcsub script command.

| | |
|--------------|--|
| RX | Received from display server. |
| TX | Transmitted to display server. |
| Bytes | Number of bytes for Display Services protocol, not TCP/IP protocol. |
| Messages | Number of messages for Display Services protocol, not TCP/IP protocol. |
| Msg. pending | RX messages pending are messages received from the TCP/IP layer, but not yet handled by the display client. TX messages pending are messages sent by the display client but not yet handled by the TCP/IP layer. |

Run-time Considerations

This section covers:

- [How to Open a Display in Run Mode](#) on page 270.
- How to interpret [Runtime Diagnostics](#) on page 270.
- Considerations for the [PC-Client](#) on page 271.
- [Color Mapping for Printing](#) on page 272.

How to Open a Display in Run Mode

To launch the display in Run mode, you can use either of the following methods:

- Hold down the CTRL key and double-click the display icon in the Object Browser.
- Select the display icon in the Object Browser, right click and choose **Run** from the context menu.



You can have Run mode and Build mode views of the same display open simultaneously.

Runtime Diagnostics

[Table 52](#) describes various symbols and messages that may occur in the Run mode.

Table 52. Runtime Diagnostics

| Message/Symbol | Description |
|---------------------------|--|
| Green Cross in Data Field | If you see a green cross in a field where you expect to see text or numeric data, it means that the client has sent a data subscription toward the network, and has not received the data back yet. This is a normal mode of operation. |
| Red Cross in Data Field | If you see a red cross in a field where you expect to see text or numeric data, it can mean either of two things, depending on the circumstances: <ul style="list-style-type: none">• If the red cross appears immediately when the display is put in Run mode, it indicates that there is an error in the dcsub statement. For instance, a misspelled word, wrong data type, and so on.• If the red cross appears in response to a data request, it may indicate that an object name has been misspelled and can not be found. |

PC-Client

Drag and Drop

To drag data from Display Services to other applications (typically Excel) point at the element to drag data from, press CTRL and left mouse button and keep them activated. Then drag the data to the position where you want to drop them and release the left mouse button.

The following elements support drag and drop (and cut/paste): Text, Numeric, Multitext, List, Edit, ComboBox, Bar, Multibar, Pie, Trend, Gauge.

Cut and Paste

To copy data from Display Services to the clipboard point at the element to copy data from, press CTRL + SHIFT and left mouse button. The element supporting copy are defined above.

Print

From the file menu it is possible to print the current display to any connected WINDOWS compatible printer (choosing **File > Print**). Select the current printer, and set it up to print. The printout will be rescaled for maximum size on the selected paper. The aspect ratio will be kept.

Color Mapping for Printing

The color map tool lets you map any display color to a different color for printing. For instance, you can change the display background color (typically a dark shade) to print as white. You can create a unique color map for each printer.

To create a color map:

1. Open a display whose colors you want to convert.
2. Launch the color map tool from the Windows taskbar. Choose: **ABB Start Menu>ABB Industrial IT 800xA>Information Mgmt>Display Services>Client> IM Display Convert Print**.
3. Use the Printer pull-down menu to specify the printer that this map is being created for, [Figure 170](#).
4. Click **New**. This displays the New Convert dialog. This dialog is used to name the color mapping specification for each color to be mapped.
5. Enter a name for the color you want to convert (for example, grey), [Figure 171](#), then click **OK**.
6. Specify the color to be converted. To do this:
 - a. Click and hold on the **From** box in the Colors area of the Convert Print Color dialog.
 - b. Drag the cursor over an area on the display that uses the color to be converted, [Figure 172](#), and then release to select the color.

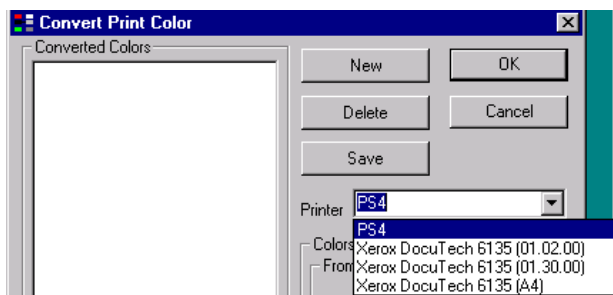
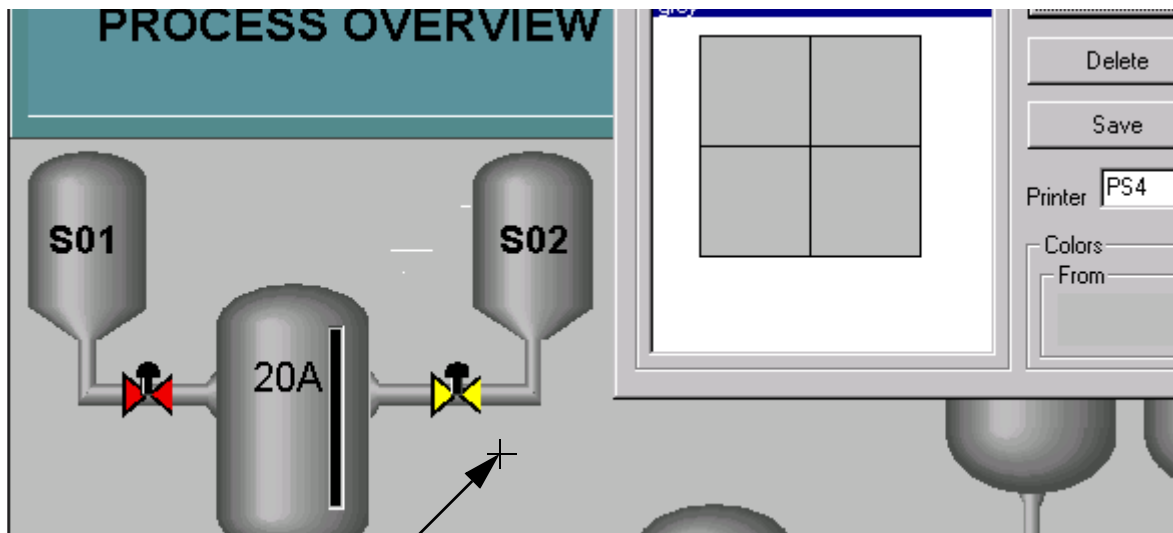


Figure 170. Selecting the Printer



Figure 171. Entering a Name for



Cursor Over Area with Color to be Converted

Figure 172. Selecting the Color to Be Converted

7. Specify the new color. To do this:

- a. Click and hold on the **To** box in the Colors area.
- b. Drag the cursor over an area on the screen that uses the new color and then release to select the new color.



You are not limited to the colors on the display. You can use any color that is currently displayed on your screen. If the color you want to use is not available, open a color palette as described in [Color](#) on page 80.

[Figure 173](#) shows a color map specification that converts grey to white.

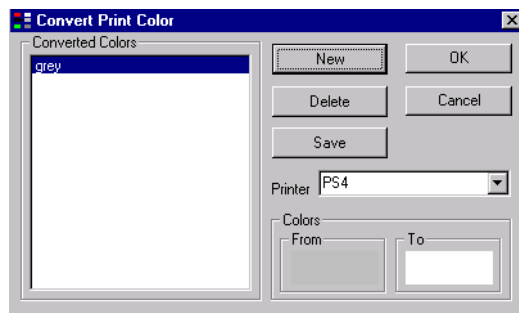


Figure 173. Converting Grey to White

8. Click **Save** to save the mapping specification for that color.
9. Repeat steps 4 through 8 for each color that needs to be converted.
10. Click **OK** when you are finished mapping all colors that need to be converted.



- If you need to create a unique map for another printer, repeat this procedure starting at step 3.
- If you have multiple printers that use the same mapping specifications, see [Working with Color Map Files](#) on page 274.

Working with Color Map Files

The color maps you create are stored as .ppc files in:

C:\Program Files\ABB Industrial IT\Inform IT\Display Services\Client\ColorMaps

The maps are automatically named after the printer they were created for. For example, the map for a printer named PS4 is PS4 .ppc, [Figure 174](#).

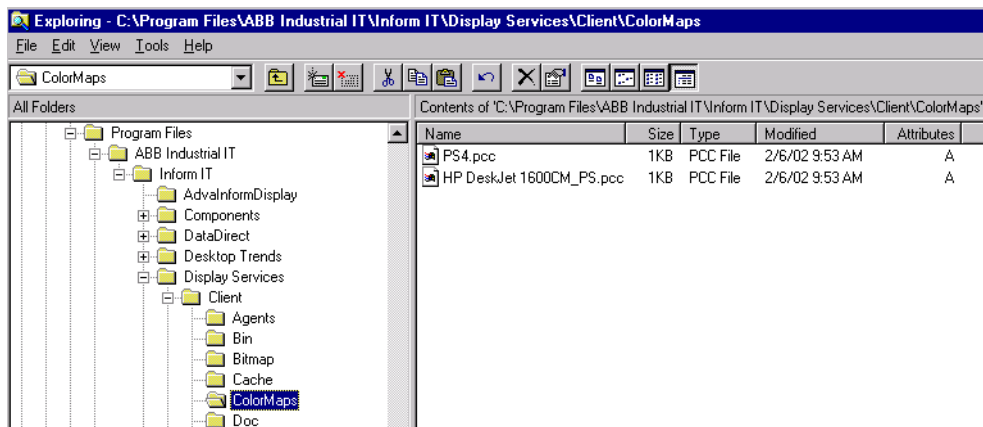


Figure 174. .pcc Files in ColorMaps Folder

You can copy these files to other display clients.

If you have more than one printer that requires the same color mapping specification, use the color mapping tool to create the color mapping specification, and then copy and rename the file for as many printers as required. For example, if you have a printer PS2 that needs the same color map as PS4, copy the PS4.pcc file and then rename the copy PS2.pcc.

Section 6 Display Scripting

Introduction

Scripts are used in Display Services to define display element properties, including queries and actions. The syntax for Display scripts is similar, but not identical to the "C" language, since Display scripts do not provide all the functionality of "C".

Since scripts are used for queries and actions in Display Services, the scripts are typically small. Data queries and attribute constants are seldom more than one line of code. Generally, only actions are longer than one line.

Display scripts support:

- simple control flow, *if-then-else-endif* for decision making and *do-while* for looping with the termination test at the bottom.
- full string handling including concatenation, substring, padding, trim, replace, search and length functions. Automatic type conversion between strings and integers/floats is also supported.
- math functions such as square root, sine, cosine and tangent.

Subroutines or functions as in "C" are NOT supported in Display scripts. Instead, you can execute a script for a specific display element from another display element.

Section Reference

This section describes the available functions and syntax rules for Display scripts. Examples are included. The information in this section is organized as follows:

- [Conventions](#) on page 278 describes the conventions used to show syntax for the various script elements.
- [General Syntax for Script Statements](#) on page 278

- [Data Types, Operators, and Expressions](#) on page 278
- [Control Flow Statements](#) on page 283
- [Queries and Actions](#) on page 285
- [Functions](#) on page 286

Conventions

The syntax descriptions in this section use the following conventions:

- *italic type* indicates a keyword that must be spelled exactly as shown
- square brackets [] are used to indicate an optional component in a statement
- <> are used to indicate programmer input
- All punctuation such as commas, parentheses, semicolons, periods, etc., must be included where shown.

For example, in the if-then-else-endif syntax description:

if <expression> *then* <statements> [*else* <statements>] *endif*;

- *if*, *then*, *else*, and *endif* are keywords
- the else clause [*else* <statements>] is optional
- <expression> and <statements> must be supplied by the programmer

General Syntax for Script Statements

An expression such as `$SessionVar.X = 12` or `print("XL300")` becomes a statement when it is followed by a semicolon. The general syntax for a statement is as follows:

```
$SessionVar.X = 12;  
$SessionVar.Y = 24;  
print("XL300", "xl300");
```

Data Types, Operators, and Expressions

Variables and constants are the basic objects manipulated in scripts. Variables are:

- Display Element properties
- Session variables
- System variables (System Variables are read only).

Operators specify what is to be done to the variables. Expressions combine variables and constants to produce new values. The type of the variable determines the set of values it can have and what operations can be performed on it.

Variable Names

Property names are pre-defined and can not be changed. Session variables are made up of letters and digits, the first character must be a letter. The underscore "_" counts as a letter. Variable names are case sensitive so "X" and "x" are two different names. Variable names can be up to 20 characters long. A variable always look as follows:

`<type><element>.<name>`

`<type>` is the data type of the variable (see below)

`<element>` is the name of the display element. For session variables the element name is always "SessionVar" and for system variables it is "SystemVar".

`<name>` is the property or variable name as explained above.

Session Variables are user defined variables which you can create and use for holding values throughout the entire Display session. Session Variables survive a change of display, so once defined they are global and accessible to all displays in the session.

Session Variables can be saved and loaded using the commands described in [Session variable](#) on page 287.

Example: Assigning an integer value to a Session Variable named MyVar

```
&SessionVar.MyVar = 38;
```

Example: Adding a constant value and a value from a Session Variable named MyVar, saving the result in a Session Variable named YourVar

```
&SessionVar.YourVar = 45 + &SessionVar.MyVar;
```

Data Types

There are four data types:

| | |
|--------------------|---|
| Integers | 32 bit integer. The type character for integers is an ampersand, "&" |
| Floats | Single precision float. The type character for floats is a hatch, "#" |
| Strings | Variable length string. The type character for strings is a dollar, "\$" |
| Multistring | Array of String. The type characters for multistrings are double dollar, "\$\$" |

Constants

Constants are integers, floats or strings.

Integers

An integer constant is defined as:

```
integer-const:
    signopt digit-sequence
sign: one of
    + -
digit-sequence:
    digit
    digit-sequence digit
digit: one of
    0 1 2 3 4 5 6 7 8 9
```

12, -45 and 1834 are all examples of legal integer constants.

As a special option, integers can be specified in time-format. Such integers are automatically converted to seconds when parsed.

Examples:

2:00 equals 2 minutes (120 seconds)

1:00:00 equals one hour (3600 seconds)

2:13:16 equals 2 hours, 13 minutes and 16 seconds (7996 seconds)

Floats

A float constant is defined as:

```
float-const:
    signopt digit-sequence . digit-sequence
sign: one of
    + -
digit-sequence:
    digit
    digit-sequence digit
digit: one of
    0 1 2 3 4 5 6 7 8 9
```

34.5, 89.0 and 251.3 are all examples of legal float constants.

Strings

A string constant is defined as:

```
string-const:
    " char-sequence "
char-sequence:
    char
    char-sequence char
char: any printable char.
```

"Beer", "Wine" and "Soda and water" are all examples of legal string constants.

If a quotation mark (") or escape (\) character is needed within the string expression, it must be preceded by an escape character (\). The escape character indicates that the next character is to be interpreted as text. This may be used when you assign a string DataQuery for an object. For example: `$Text_1.DataQuery = "|" + hex(&Level.Value) + "|"`;

If `hex(&Level.Value)` returns the value 4f, the `Text_1 DataQuery` property will contain the value "4f". The double quotes are needed since a query starting with a number, is not recognized as a string.

Arithmetic Operators

Arithmetic operators, [Table 53](#), apply to integers or/and floats. The addition operator (+) is also for concatenating strings.

Table 53. Arithmetic Operators

| Operator | Operation | Supported Datatypes |
|----------|--------------------------|--------------------------|
| + | Addition (concatenation) | Integer,Float and String |
| - | Subtraction | Integer and Float |
| * | Multiplication | Integer and Float |
| / | Division | Integer and Float |
| % | Modulus | Integer |

Relational and logical Operators

The relational and logical operators are listed in [Table 54](#).

Table 54. Relational and logical Operators

| Operator | Operation | Supported Datatypes |
|----------|--------------------------|---------------------------|
| > | Greater than | Integer, Float and String |
| >= | Greater than or equal to | Integer, Float and String |
| < | Less than | Integer, Float and String |
| <= | Less than or equal to | Integer, Float and String |
| = | Equal to | Integer, Float or String |
| != | Not Equal to | Integer, Float or String |
| and | Logical AND | Integer |
| or | Logical OR | Integer |

Bitwise Operators

Bitwise operators supported in Display scripts are listed in [Table 55](#).

Table 55. Bitwise Operators

| Operator | Operation | Supported Datatypes |
|----------|-------------|---------------------|
| & | Bitwise AND | Integer |
| | Bitwise OR | Integer |

Type Conversions

Type conversions are done automatically between integer, float and strings. It is not possible to convert to or from a multistring. When converting from string to integer or float, the string must contain the number in literal form. If the string is non-convertible (for example: "hello"), the converted value is unpredictable.

Comments

A comment is a sequence of non-executable characters following a pair of slashes(*//*). The rest of a line after a pair of slashes are treated as a comment. Comments can be used any place in a script to describe the code. Example of a comment:

```
// Following code will change the shape of
// the shape "Shape_1" to indicate alarm
$Shape_1.Shape = "Box";
..
```

Control Flow Statements

The control flow statements specify the order in which computations are performed.

If-Then-Else-Endif

The if-then-else-endif statement is used to make decisions. The syntax is:

```
if <expression> then <statements> [else <statements>] endif;
```

Note that the *else* part is optional. Nesting If-Then-Else-Endif is limited to 250.

The expression (type must be integer) is evaluated. If it is true (not 0) the statements specified between *then* and *endif* (or *else*) are computed. If the else part is included, the statements between *else* and *endif* are computed if the expression are false (0).

Example 1:

```
if &Edit_1.Visible then
    $Text_1.Foreground = $Edit_1.Value;
    update("Text_1");
endif;
```

Example 2:

```
if #SessionVar.ScanCount = 1.0 then
    #ScopeAverageWeight.DataQuery = #M_6.Value[2,6];
    #SessionVar.ScanCount = #SessionVar.ScanCount + 1.0;
else
    if #SessionVar.ScanCount = 2.0 then
        #ScopeAverageWeight.DataQuery = #M_6.Value[3,6];
        #SessionVar.ScanCount = #SessionVar.ScanCount + 1.0;
    else
        if #SessionVar.ScanCount = 3.0 then
            #ScopeAverageWeight.DataQuery = #M_6.Value[4,6];
            #SessionVar.ScanCount = #SessionVar.ScanCount + 1.0;
        else
            if #SessionVar.ScanCount = 4.0 then
                #ScopeAverageWeight.DataQuery = #M_6.Value[5,6];
                #SessionVar.ScanCount = 1.0;
            endif;
        endif;
    endif;
endif;
```

Do-While

The Do-While statement is used to produce a loop. The syntax is:

do; <statements> *while* <expression>;

Note the semicolon after *do*. The statements between *do* and *while* are computed until the expression is false (0). This means the statements are always computed once.

Nesting Do-While is limited to 250

The following example shows how to dump a list to a file using the do-while statement.

```
&SessionVar.Counter = 0;
system("echo |"|" > log.file");
do;                                     // Do not forget the ';'
    $SessionVar.OneLine = $List_1.Value[&SessionVar.Counter];
    system("echo |"+"$SessionVar.OneLine+"|" >> log.file");
    &SessionVar.Counter = &SessionVar.Counter +1;
while ($SessionVar.OneLine != "");
```

Queries and Actions

Queries

A query is a script that feeds data to a display element. A query can describe data received synchronously or asynchronously. Synchronous data may be SQL statements, constants or variables. Asynchronous queries are typically OCS requests where data are sent from the OCS components (Controllers, OS or IMS) cyclically or on event. Queries can only contain the functions described in [Table 58](#), a constant value or reference properties in other objects.

ReQueries

The ReQuery is an action that re-executes a query when the object to be queried has changed (for instance to re-query data for a List or a Combobox, or to change a DCS subscription in a DataQuery at runtime). The syntax for the ReQuery uses the script command 'execute'. For example, to perform the ReQuery of a TextObject called MyText: `execute("MyText", "ReQuery")`;

The ReQuery action is applicable for all display element types that have a DataQuery property. These display element types are: Bar, Combobox, Edit, Gauge, List, MultiBar, MultiText, Numeric, Pie, Text, Matrix, and XYPlot.



When the data source for a query is a Matrix display element, the execute requery action must be followed by an Update action. For example, if the data query for a numeric display element is: `$Matrix_1.Value[1,2]` ;, then the script to get new data for the numeric element would be:

```
execute ("Numeric_1","ReQuery");
update (Numeric_1);
```

Actions

An action in Display Services is a script that executes on a specified event. These events are typically when a push button is activated or when data in a field changes. The *Display Services User's Guide* provides a listing of properties for each type of display element.

Actions can be quite complex (several lines of code). Actions let you change the behavior and appearance of the entire display by changing the properties of the display elements.

Actions can contain all types of functions and can be used for both reading values from other objects or setting properties in other objects.

Functions

Functions are grouped as: [System variable](#), [Session variable](#), [Query](#), [Action](#), [String](#), [Math](#), and [General Purpose](#).

System variable

System variable functions access OS generated system variable files, [Table 56](#).

Table 56. System Variable functions

| Function | Description | Section |
|----------|-----------------------|------------------------------------|
| lsyvar | Load System Variables | lsyvar on page 311 |

Session variable

Session variables, [Table 57](#), are related to a Display building session. The variables can be defined, loaded and saved. The variables are saved in the server using the username as ID.

Table 57. Session Variable functions

| Function | Description | Section |
|----------|--------------------------------------|------------------------------------|
| dsevar | Define session variable | dsevar on page 305 |
| isevar | Check if session variable is defined | isevar on page 309 |

Query

Query functions are used in query statements, [Table 58](#). See [Queries](#) on page 285.

Table 58. Query functions

| Function | Description | Section |
|---------------|--|---|
| dcssub | Subscribe from OCS | dcssub on page 300 |
| dcsgetobjinfo | Query object info from OCS | dcsgetobjinfo on page 300 |
| sql | Query data from ORACLE | sql on page 317 |
| asql | Query data asynchronously from ORACLE | asql on page 291 |
| data | The data function lets you perform one of the following: <ul style="list-style-type: none">• modify an entry for a numeric history log• add a new entry to a numeric history log• write to a process object• read from or write to an OPC object• subscribe to the last history value | data on page 295 |
| group | Returns a list of defined display groups | group on page 307 |
| columns | Concatenate column data, return as multistring | columns on page 294 |
| file | Read an ASCII file | file on page 306 |

Action

Action functions, [Table 59](#), are used in action statements ([Actions](#) on page 286).

Table 59. Action functions

| Function | Description | Section |
|-------------|-------------------------------------|---|
| exit | Exit the Display session | exit on page 306 |
| print | Print current display | print on page 313 |
| display | Change display | display on page 303 |
| opendisplay | Open a second display in MDI mode | opendisplay on page 312 |
| basedisplay | Open a display in base display area | basedisplay on page 292 |
| close | Close current display in MDI mode | close on page 294 |
| system | Execute a system call | system on page 319 |
| update | Update a display element | update on page 323 |
| execute | Execute a display element method | execute on page 305 |
| sql | Execute a SQL statement | sql on page 317 |
| insert | Element type dependent insertion | insert on page 308 |
| delete | Element type dependent deletion | delete on page 302 |

String

String functions, [Table 60](#), are used to manipulate strings.

Table 60. String functions

| Function | Description | Section |
|----------|-----------------------------------|------------------------------------|
| length | Examine length of a string | length on page 310 |
| substr | Extract a substring from a string | substr on page 318 |
| rtrim | Trim a string to the right | rtrim on page 316 |
| ltrim | Trim a string to the left | ltrim on page 311 |

Table 60. String functions

| Function | Description | Section |
|----------|---|-------------------------------------|
| replace | Replace characters in a string | replace on page 315 |
| instr | Position of a substring in a string | instr on page 309 |
| lpad | Pad a string to the left | lpad on page 311 |
| rpadd | Pad a string to the right | rpadd on page 315 |
| upper | Convert all characters in a string to uppercase | upper on page 324 |
| lower | Convert all characters in a string to lowercase | lower on page 310 |
| ascii | The ASCII value of a character | ascii on page 291 |
| chr | Character with ASCII value | chr on page 293 |
| dec | Convert a numeric to string | dec on page 302 |
| hex | Convert numeric to string in hexadecimal. | hex on page 308 |

Math

Math functions are listed in [Table 61](#).

Table 61. Math functions

| Function | Description | Section |
|----------|---------------------------------------|----------------------------------|
| sqrt | Square root of a numeric | sqrt on page 318 |
| sin | Sine of a numeric (in radians) | sin on page 317 |
| asin | Arc sine of a numeric (in radians) | asin on page 291 |
| cos | Cosine of a numeric (in radians) | cos on page 294 |
| acos | Arc cosine of a numeric (in radians) | acos on page 290 |
| tan | Tangent of a numeric (in radians) | tan on page 319 |
| atan | Arc tangent of a numeric (in radians) | atan on page 292 |

General Purpose

General Purpose functions are listed in [Table 62](#).

Table 62. General purpose functions

| Function | Description | Section |
|-------------|--|---|
| bit | Test for a bit in an expression | bit on page 292 |
| oscolor | Convert an Operator Station color | oscolor on page 312 |
| osfont | Convert an Operator Station font | osfont on page 312 |
| rand | Random number | rand on page 314 |
| time | Time conversion | time on page 320 |
| ishost | Check for client machine type | ishost on page 310 |
| clientinfo | Provides positioning information for positioning user element on the screen. | clientinfo on page 293 |
| ostreatment | Convert Operator Station treatment | ostreatment on page 313 |
| getenv | Get environment variable | getenv on page 307 |
| getpref | Get the name of the currently logged in user. | getpref on page 307 |
| trace | Print trace messages | trace on page 323 |
| dialog | Display Information, Warning or Error dialog | dialog on page 303 |
| group | Get all defined groups | group on page 307 |

acos

The **acos** function calculates the arc cosine of a specified *number* into a float. The angle is given in radians.

Syntax **float** *acos*(**numeric-expr** <number>)

Apply to Expressions

Return value The arc cosine value.

Example &Numeric_1.DataQuery = 12 + acos(0.4);
 update("Numeric_1");

ascii

The **ascii** function returns the ASCII value of the first character in a *string*. If the string is empty it returns 0.

| | |
|--------------|--|
| Syntax | integer <i>ascii</i> (string-expr <string>) |
| Apply to | Expressions |
| Return value | The ASCII value. |
| Example | <pre>&Numeric_1.DataQuery = ascii("A"); update("Numeric_1");</pre> |

asin

The **asin** function calculates the arc sine of a *number* into a float. The angle is given in radians.

| | |
|--------------|--|
| Syntax | float <i>asin</i> (numeric-expr <number>) |
| Apply to | Expressions |
| Return value | The arc sine value. |
| Example | <pre>#Numeric_1.DataQuery = 12.1 + asin(0.5); update("Numeric_1");</pre> |

asql

The **asql** function queries data from ORACLE asynchronously.

| | |
|--------------|---|
| Syntax | <i>asql</i> ([integer <channel>], string-expr <sql-expression>) <channel> is an optional parameter for referencing a specific data provider when there are more than one data provider of the same type. If <channel> is not specified, the function will access the default data provider which is CHANNEL 0. For the <i>sql-expression</i> , use the form: <code>select ...</code> |
| Apply to | DataQuery in the Matrix element only |
| Return value | None. |
| Example | <pre>asql("select * from all_lines")</pre> |

atan

The **atan** function calculates the arc tangent of a *number* into a float. The angle is given in radians.

| | |
|--------------|--|
| Syntax | float <i>atan</i> (numeric-expr <number>) |
| Apply to | Expressions |
| Return value | The arc tangent value. |
| Example | <pre>&Numeric_1.DataQuery = 12 + atan(0.4); update("Numeric_1");</pre> |

basedisplay

The **basedisplay** function opens a specified display in the base display area, if the base display preference is enabled. This function is similar to the [display](#) function, [display](#) on page 303.

| | |
|--------------|---|
| Syntax | void <i>basedisplay</i> (string-expr <group-name>, string-expr <display-name>) |
| Apply to | Action |
| Return value | None |
| Example | <pre>basedisplay("DisplayDemo","BaseDisp");</pre> |

bit

The **bit** function returns the value for a specified bit in an *expression*.

| | |
|--------------|--|
| Syntax | integer <i>bit</i> (integer-expr <expression>, integer-expr <bit>) |
| Apply to | Expressions |
| Return value | The bit value (0 or 1). |
| Example | <pre>&Numeric_1.DataQuery = bit(&Numeric_2.Value,12); update("Numeric_1");</pre> |

broadcast

This broadcasts a string to all clients connected to the server. When a client receives the broadcast the action Broadcast is called. The property Message holds the broadcasted message.

| | |
|--------------|--|
| Syntax | <i>broadcast</i> (string-expr <string>) |
| Apply to | Actions |
| Return value | None. |
| Example | <code>broadcast("System Administration begins in 5 minutes");</code> |

chr

The **chr** function returns a one-character string with the character of the ASCII value of a specified *number*.

| | |
|--------------|---|
| Syntax | string <i>chr</i> (integer-expr <number>) |
| Apply to | Expressions |
| Return value | String with character of the ASCII value. |
| Example | <code>&Numeric_1.DataQuery = chr(47); update("Numeric_1");</code> |

clientinfo

The **clientinfo** function returns the screen resolution.

| | |
|--------------|---|
| Syntax | <type-char> <i>clientinfo</i> (string-expr <infotype>) <Infotype> can be: XRES - returns X-screen resolution as an integer. YRES - returns Y-screen resolution as an integer. For PC-clients these are the values selected in the Display Setup program. |
| Apply to | Expressions |
| Return value | String with character of the ASCII value, for example 1280 or 1024 |

Examples &Numeric_1.DataQuery = &clientinfo("XRES");
 &Numeric_1.DataQuery = &clientinfo("YRES");

columns

columns returns columns concatenated (optionally formatted) into a string array.

Syntax \$\$*columns*(**string-expr** <matrix>, **column-expr** <column>)

Description The *column expression* has the following format:
 [**integer-expr** <column-no>] or
 [**integer-expr** <column-no>,**string-expr** <format>]
 The format syntax is: <justify><width>.<precision>
 justify options are l=left, c=center, r=right.

Apply to String array queries

Return value Array of formatted columns.

Example \$\$columns(Matrix_1,[1,"r20"],[2,"r10.5"]);

cos

The **cos** function calculates the cosine of a *number* into a float. The angle is given in radians.

Syntax **float** *cos*(**numeric-expr** <number>)

Apply to Expressions

Return value The cosine value.

Example &Numeric_1.DataQuery = 12 + cos(0.4);
 update("Numeric_1");

close

The close function closes the current display when multiple displays are open in the MDI run mode. Close is not applicable when operating in the SDI run mode.

Syntax *close*()

Apply to Action

Return value None.

Example `close();`

data

The **data** function lets you perform one of the following:

- read from or write to an OPC object
- modify or add an entry for an asynchronous (lab data) numeric history log. For instructions on how to configure a lab data log refer to *System 800xA Information Management Data Access and Reports (3BUF001094*)*.
- write to an Advant process object
- subscribe to the last history value

The data statement is applied in the DataQuery property of a Matrix display element. You may also use the data statement in a Numeric or Text display element to read OPC data.

When you use the data statement in a Matrix display element, if the application requires a value to be displayed, a visual element such as a text or numeric element is required to display the value returned. Status information related to the data statement is stored in, and can be read from the Matrix element.

In the matrix you can either use the event datachange property to tell the visual element to refresh, or have the visual element subscribe to the matrix.

For details regarding the data statement applications listed above, refer to:

- [To Read from an OPC Object](#) on page 296
- [Authority for Write Access to History Logs and Process Objects](#) on page 296
- [To Modify an Entry in an Asynchronous Numeric \(Lab Data\) Log](#) on page 297
- [To Add a New Entry in an Asynchronous Numeric \(Lab Data\) Log](#) on page 297
- [To Write to an Advant Process Object](#) on page 298
- [To Write to an OPC Object](#) on page 299
- [To Subscribe to the last History Value for an Object](#) on page 299

To Read from an OPC Object

| | |
|--------------|---|
| Syntax | <p><code>data(string <name>, "Subscribe", string <object name>, string <Subscription rate>, [string <OPC Access Path>])</code></p> <p><name> is used to access a specific data provider, typically OPC or AIPOPC).</p> <p><object-name> can be the valid name of an OPC object.</p> <p><subscription rate > is one of the following:</p> <ul style="list-style-type: none"> – dcs_DEMAND: Value is read once – dcs_EVENT: Value is read once, and then when value changes. – dcs_1s: Value is read each second. – dcs_3s: Value is read each third second. – dcs_9s: Value is read each ninth second. <p><OPC access path> is an optional string parameter. It is used as OPC Access path if specified. For example a remote object is accessible with ItemID <Obj1>, but it can also be accessed through different links, high speed or public line. Specifying the latter will be a typical use of AccessPath.</p> |
| Apply to | Dataquery |
| Return value | None. Data are sent to the display element asynchronously. |
| Example | <code>data("OPC", "Subscribe", "OPCobj", "dcs_DEMAND")</code> |

Authority for Write Access to History Logs and Process Objects

There are two levels of authority for all *Write* operations for history logs and process objects - level 1 is associated with the applicable data provider, and level 2 is associated with the user.

The ADSdpLOG, ADSdpOCS, and ADSdpOPC data providers have a parameter that determines whether or not clients can write to the objects. The user authority is stored in the user preference file. Refer to the *System 800xA Information Management Configuration (3BUF001092*)* for details on configuring user preferences.

Data provider authority has precedence over the *user* level.

To Modify an Entry in an Asynchronous Numeric (Lab Data) Log

| | |
|--------------|---|
| Syntax | <p><code>data(string <name>, "ModifyLM", string<log name>, Float <value>, Integer <Log time>, integer <Log time milliseconds>, string-expr <Log entry object status>)</code></p> <p><name> is used to access a specific data provider. The default name for the data provider that supports this function is LOG.</p> <p><log name> can be any valid name of a history log managed by Information Manager History Services software. The full log name (not just the access name) must be used.</p> <p><Log time> Enter as seconds since 1-1-1970 (UTC). Determine the time stamp of the entry you want to modify, then use the time function to compute the number of seconds. The time function is only supported on the PC platform.</p> <p><Log time milliseconds> Enter a value of 0 (zero).</p> <p><Log entry object status> Enter a value of 0 (zero) to get the actual status of the log entry, or enter a value of n to set the status to BAD. The status is accessible via the Matrix display element. For details refer to Matrix on page 131.</p> |
| Apply to | Dataquery |
| Return value | None. Data are sent to the display element asynchronously. |
| Example | <code>data("LOG", "ModifyLM", "\$HSAITEST,VALUE-1-o", 11.05, 959171516, 0, 0)</code> |

To Add a New Entry in an Asynchronous Numeric (Lab Data) Log

The syntax for adding a new entry to an existing log is the same as for modifying a log entry, except that the keyword *AddNLog* is used in place of *ModifyLM*.

| | |
|---------|---|
| Example | <code>data("LOG", "AddNLog", "\$HSAITEST,VALUE-1-o", 11.05, 959171516, 0, 0)</code> |
|---------|---|

To Write to an Advant Process Object

Syntax *data(string <name>, "Write", string <object name>, string <object type>, string <object attribute>, Float/Integer/String <value>)*

 <name> is used to access a specific data provider. The default name for the data provider that supports this function is DCS.

 <object-name> can be the valid name of an Advant object available on a node connected to the Advant OCS.

 <object-type> indicates the object type of the specified <object-name>, for example: "AI", "DI", and so on. See [Table 63](#).

 <object-attribute> defines the attribute of the object to subscribe. ("VALUE", "DESCRIPTION" etc.)

Apply to Dataquery

Example *data("DCS", "Write", "AITEST", "AI", "VALUE", 105.05)*

Table 63. Supported⁽¹⁾ Objects

| Object Type | Attributes |
|-------------------------------------|--|
| AI | VALUE, HI_LIM1, HI_LIM2, LO_LIM1, LO_LIM2 |
| AO | VALUE |
| DI and DO | STATUS (Write 0/1 - sets bit 8) |
| DAT | R_VAL, IL_VAL, IW_VAL, B0_VAL ⁽²⁾ |
| TEXT | TEXT |
| PIDCON/PIDCONA ⁽³⁾ | MMI_SP |
| MOD 300 Object Types ⁽⁴⁾ | - |

- (1) These objects are generally supported; however, there may be cases due to special uses of data types and bit strings where these objects will not work.
- (2) To write Boolean, use B0_VA to write to bit 0; otherwise, use IL_VAL to write all 32 bits.
- (3) For systems with Master software only
- (4) All MOD 300 Object Types with a single value property are supported.

To Write to an OPC Object

The syntax for writing to an OPC object is the same as for writing to any other process object, except that object type and attribute are not specified.

Example `data("OPC", "Write", "OPCTEST", 105.05)`

To Subscribe to the last History Value for an Object

There are two forms for this function - one retrieves just the value, the other retrieves the value and time stamp. The following examples illustrate:

Value Only: `data("DCS", "SubscribeLHV", "AITEST", "VALUE")`

Value & Timestamp: `data("DCS", "SubscribeLHVex", "AITEST", "VALUE")`

Syntax `data(string <name>, "SubscribeLHV<ex>", string <object name>, string <object attribute>)`

<name> is used to access a specific data provider.

<object-name> valid name of any Advant object available on a node connected to the Advant OCS.

<object-attribute> defines the attribute of the object to subscribe. ("VALUE", "DESCRIPTION" etc.)

Apply to Dataquery

Return value None. Data are sent to the display element asynchronously.

Example `data("DCS", "SubscribeLHV", "AITEST", "VALUE")`

When a numeric element pulls data from a matrix element it should display a red cross when data is no longer available. Instead it displays the last known value. To fix this, set up the DataChange event for a matrix element with the following script. This example assumes the element name Numeric_1 will get the data:

```
if (&This.DataStatus = 0) then
    // Matrix status OK, copy value to numeric element
    &Numeric_1.DataQuery = &This.Value[1,1];
    execute(Numeric_1, ReQuery); // To go from bad to good
    update(Numeric_1);
else
```

```
// Matrix status bad
if (&Numeric_1.DataStatus = 0) then
    // Numeric element not already bad (avoid error trace)
    // Set numeric element in error
    $Numeric_1.DataQuery = "Bad Numeric, (Anything goes)";
    execute(Numeric_1, ReQuery);
    update(Numeric_1);
endif;
endif;
```

dcsgetobjinfo

dcsgetobjinfo returns the object type for the specified object name.

Syntax **void** <**type-char**>dcsgetobjinfo([**integer** <channel>,]
 string-expr <object-name>, *objType*)

<type-char> indicates the information's **data type**.

<channel> is an optional parameter for referencing a specific data provider when there are more than one data provider of the same type. If <channel> is not specified, the function will access the default data provider which is CHANNEL 0.

<object-name> valid name of any Advant object available on a node in the Advant OCS.

objType specifies object type as the requested information.

| Apply to | Dataquery |
|----------|-----------|
|----------|-----------|

| | |
|--------------|--|
| Return value | None. Data are sent to the display element asynchronously. |
|--------------|--|

```
Example // Get the object type of "B0502"
$dcsgetobjinfo("B0502","objType")
```

dcssub

The **dcssub** function subscribes a specified *object-attribute* of a specified *object-name* of a specified *object-type*. The function uses the access method as specified in *access-method*.

| | |
|--------------|---|
| Syntax | <p>void< type-char><i>dcssub</i> ([integer <channel>, string-expr<object-name>, string-expr <object-type>, string-expr <object-attribute>, <access-method>)</p> <p><type-char> indicates the data type for the subscribed data.</p> <p><channel> is an optional parameter for referencing a specific data provider when there are more than one data provider of the same type. If <channel> is not specified, the function will access the default data provider which is CHANNEL 0.</p> <p><object-name> name of an object available on Advant OCS node.</p> <p><object-type> indicates the object type of the specified <i>object-name</i>. ("AI", "DI" etc).</p> <p><object-attribute> defines the attribute of the object to subscribe. ("VALUE", "DESCRIPTION" etc.).</p> <p>The following access methods are legal:</p> <p><i>dcsub_demand</i> - Data subscribed and read only once.</p> <p><i>dcsub_1s</i> - Data subscribed and read, and updated every second.</p> <p><i>dcsub_3s</i> - Data subscribed and read, and updated every third second.</p> <p><i>dcsub_9s</i> - Data subscribed and read, and updated every ninth second.</p> <p><i>dcsub_event</i> - Data subscribed and read, and updated on event.</p> |
| Apply to | Dataquery |
| Return value | None. Data are sent to the display element asynchronously. |
| Example | <pre>.. // Get the description from "B0502" \$dcsub("B0502","AI","DESCRIPTION",dcsub_demand) ..</pre> |

The dcssub function also supports bit-level access of integer attributes. This is limited to the first 28 bits of the integer value. The syntax for this application of dcssub is:

| | |
|---------|--|
| Syntax | void < type-char > <i>dcssub</i> ([integer <channel>, string-expr <object-name>, string-expr <object-type>, string-expr <object-attribute>: integer <bit>, <access-method>) |
| Example | <code>\$dcssub("B0502","DAT","VAL_TYPE:12",dcs_demand)</code> |

dec

The **dec** function converts a specified *value* to a string, in a specified *format*. If no format is specified, the default is %d. The syntax of the format follows the C-function `printf()` syntax.

| | |
|--------------|--|
| Syntax | string <i>dec</i> (numeric-expr <value> [, string-expr <format>]) |
| Apply to | Expressions, |
| Return value | The converted string |
| Example | .. \$AI1Value.DataQuery = " " + dec(#AI1.Value,"%0.2f") + " "; update("AI1Value"); .. |

delete

This deletes a specified row from a matrix element.

| | |
|--------------|--|
| Syntax | void <i>delete</i> (string-expr <element-name>, string-expr <property>, numeric-expr <number>) <element-name> must be a matrix element name. <property> must be specified as "Row". <number> indicates the row number, where 1 is the first row. -1 deletes the last row. |
| Apply to | Actions |
| Return value | None. |

```

Example      ..
              delete(Matrix_1,"Row",5);
              ..

```

dialog

The **dialog** function displays an information, warning or error dialog depending on the *severity* level. The caption is set to *caption* and text is set to *text*, The dialog has a button to confirm the message.

Syntax **void** *dialog*(**string-expr** <severity>, **string-expr** <caption>, **string-expr** <text>)

For <severity> enter: "Information", "Warning" or "Error".

Apply to Actions

Return value None

```

Example      ..if (oraerr() = 1403) then
              dialog("Information","ORACLE Access","No records found");
              endif;..

```

display

The **display** function can be used in two ways:

Function 1: clears the current display and loads a different display as specified by *display-name* and *group-name*. See also [basedisplay](#) on page 292 and [opendisplay](#) on page 312.

Syntax **void** *display*(**string-expr**<groupname>, **string-expr** <displayname>)

Apply to Push button Actions

Return value None

```

Example      // When pushing the button "Menu" load the main menu display
              display("MAIN","MENU");

```

Function 2: returns a list of defined displays in a specified group.

| | |
|--------------|--|
| Syntax | <code>\$\$display(string-expr<groupname>[, integer-expr <include-description>])</code> |
| Description | The optional parameter <include-description> indicates whether or not the display description is added to the display name. The default for <include-description> is 0 (means display name only). If the <include-description> is set to 1, each string in the returned list will have the following format: < display name (22 chars)><description> |
| Apply to | Dataquery |
| Return value | The list |
| Example | The List_2 shall reflect all displays in the List_1.SelectedValue Following will be the DataQuery for List_2 <code>\$\$display(\$List_1.SelectedValue,0);</code> |

displayexist

The **displayexist** function checks for existence of a specific display.

| | |
|--------------|--|
| Syntax | <code>integer displayexist(string-expr <group-name>,string-expr <display-name>)</code> |
| Apply to | Expressions |
| Return value | 1 If the display exist, 0 otherwise |
| Example | <code>.. if (displayexist("START","STARTER") then display("START","STARTER"); endif; ..</code> |

dlovar

The **dlovar** function defines a local variable with a specified *Name*, and initializes it with a specified *InitialValue*.



Local variables created with `dlovar` are local to the display element where they occur.

- If you need a local variable which is local to the display use user properties.
- If you need a local variable which is local to the session use session variables.

The local variables are accessed by the Local prefix.

| | |
|--------------|--|
| Syntax | <code>dlovar(string-expr <Name>, string-expr <InitialValue>)</code> |
| Apply to | Actions |
| Return value | None |
| Example | <pre>dlovar("CurrentLog",\$Edit_1.Value); dlovar("CurrentAttribute",\$Edit_2.Value); \$Trend_1.Object_1 = \$Local.CurrentLog; \$Trend_1.Attrib_1 = \$Local.CurrentAttribute;</pre> |

dsevar

The **dsevar** function defines a session variable with a specified *Name* and initializes it with a specified *InitialValue*. If the variable is already defined, it is set to *InitialValue*.

| | |
|--------------|--|
| Syntax | <code>dsevar(string-expr <Name>, string-expr <InitialValue>)</code> |
| Apply to | Actions |
| Return value | None |
| Example | <pre>dsevar("CurrentLog",\$Edit_1.Value); dsevar("CurrentAttribute",\$Edit_2.Value); \$Trend_1.Object_1 = \$SessionVar.CurrentLog; \$Trend_1.Attrib_1 = \$SessionVar.CurrentAttribute;</pre> |

execute

The **execute** function calls and executes a specified *action* in a specified *element*. The function can be used to call an action like calling a subroutine. DO NOT execute a ReQuery action within an OnPreQuery action.

| | |
|--------------|--|
| Syntax | void <i>execute</i> (string-expr <element-name>, string-expr <action-name>) Action-name can be any display element property of type ACTION, or ReQuery. |
| Apply to | Actions |
| Return value | None |
| Examples | // Call the "Action" action in a push-button execute("Push_1","Action"); //Execute a requery on MultiText_1 execute("Multitext_1","ReQuery"); |

exit

The **exit** function terminates the current Display session.

| | |
|--------------|-----------------|
| Syntax | <i>exit</i> () |
| Apply to | Actions |
| Return value | None. |
| Example | <i>exit</i> (); |

file

The **file** function returns the contents of the specified text file (15 KB maximum), and puts it in the DataQuery property in a List, Multitext, or Combobox element.

| | |
|--------------|---|
| Syntax | <i>\$file</i> ([integer <channel>], string-expr <file-name>) <channel> is an optional parameter for referencing a specific data provider when there are more than one data provider of the same type. If <channel> is not specified, the function will access the default data provider which is CHANNEL 0. The <i>file-name</i> can be any valid name of an available file. |
| Apply to | Dataquery in List, Multitext, or Combobox element |
| Return value | The returned value as a string. |

Example `$List_1.DataQuery = $file("etc/hosts");
update("List_1");`

getenv

The **getenv** function returns the value of the specified *environment-variable*. If the variable does not exist, an empty string is returned.

Syntax **string** *getenv*(**string-expr** <environment-variable>)

Apply to Expressions

Return value The returned value as a string.

Example `$Text_1.DataQuery = "I"+getenv("USER")+"I";
update("Text_1");`

getpref

The **getpref** function accesses the preference file and returns the currently logged in user.

Syntax **string** *getpref*("USER")

You can also get the name of the user currently logged in to the PC:

Syntax **string** *getpref*("OS-USER")

group

The **group** function returns a list of defined display groups.

Syntax **multistring** *\$\$group*()

Apply to Dataquery

Return value The list

Example The List_1 shall reflect all display groups
Following will be the DataQuery for List_1
`$$group();`

hex

The **hex** function converts a specified *number* into a string in hexadecimal notation.

| | |
|--------------|---|
| Syntax | string <i>hex</i> (integer-expr <number>) |
| Apply to | Expressions |
| Return value | The converted string |
| Example | <code>\$DI1Status.DataQuery = "I"+hex(&Status.Value)+"I"; update("DI1Status");..</code> |

iif

The **iif** function returns one of two expressions depending on the test-expression. The two supplied expressions must be of the same type as specified by the type character in front of the iif function. The iif function is mainly for data queries since the if-then-else statement apply to actions only.

| | |
|--------------|--|
| Syntax | <type-char> <i>iif</i> (integer-expr <test-expr>, <div style="margin-left: 100px;">expr <iftrue-expr>, expr <iffalse-expr>)</div> <div style="margin-left: 100px;"><type-char> indicates the data type for the expression returned.</div> |
| Apply to | Expressions. |
| Return value | One of the supplied expressions |
| Example | <code>.. // if any ORACLE error return the number else empty string \$iif(&oraerr(),\$oraerr,"")</code> |

insert

This inserts a row in a matrix element.

| | |
|--------|--|
| Syntax | void <i>insert</i> (string-expr <element-name>, string-expr <property>, numeric-expr <number>) <div style="margin-left: 100px;"><element-name> must be a matrix element name. <property> must be specified as "Row".</div> |
|--------|--|

<number> indicates the row number, where 1 is the first row.
-1 appends a row after the last row.

Note: The matrix element must contain at least a row of data retrieved by an ASQL statement.

| | |
|--------------|---------------------------|
| Apply to | Actions |
| Return value | None. |
| Example | insert(Matrix_1,"Row",0); |

instr

The **instr** function returns the position of the *occurrence-no*th occurrence of *pattern-string* in *source-string* beginning search at position *begin-pos*. Position is relative to the first character in *source-string*.

| | |
|--------|---|
| Syntax | integer <i>instr</i> (string-expr <source-string>, string-expr <pattern-string> [, integer-expr <begin-pos> [, integer-expr <occurrence-no>]]) |
|--------|---|

Begin-pos is set to 0 and *occurrence-no* is set to 1 if omitted.

| | |
|--------------|---|
| Apply to | Expressions. |
| Return value | The position |
| Example | // Return the position of "apple" &SessionVar.pos = instr("orangebananaapplepeach","apple");.. |

isevar

The **isevar** function searches for a session variable with a specified *Name*. If the variable is defined the function returns true, otherwise false.

| | |
|--------------|---|
| Syntax | integer <i>isevar</i> (string-expr <Name>) |
| Apply to | Actions |
| Return value | 1 for defined variable, 0 for not. |
| Example | if isever("CurrentLog") then \$Trend_1.Object_1 = \$SessionVar.CurrentLog; |

```
else  
$Trend_1.Object_1 = $Edit_1.Value;  
endif;
```

ishost

The **ishost** function checks whether or not the host is the specified type.

Syntax **integer** *ishost*(**string-expr** <host-type>)

Host-type can be "WIN" or "UNIX".

Apply to Expressions

Return value 1 if host is the specified *host-type*, 0 otherwise.

Example if (ishost("UNIX")) then
 system("hpterm&");
 endif;

length

The **length** function returns the length of the specified *string* in bytes.

Syntax **integer** *length*(**string-expr** <string>)

Apply to Expressions

Return value The number of bytes in the *string* as integer

Example &SessionVar.Total = 123 + length(\$Edit_1.Value);

lower

The **lower** function returns a *string* with all characters converted to lowercase.

Syntax **string** *lower*(**string-expr** <string>)

Apply to Expressions

Return value The converted string.

Example \$SessionVar.lwc = lower("Abdominal");

Ipad

The **lpad** function pads the *string* to the specified *number* of characters.

Syntax

```
string lpad(string-expr <string>,  
            integer-expr <number> [,  
            string-expr <padder>])
```

The *padder* character is used for padding. If *padder* is omitted a space ' ' is used for padding.

Apply to Expressions.

| | |
|--------------|-------------------|
| Return value | The padded string |
|--------------|-------------------|

```
Example // Pad a string to 20 in length. Use the '*' as padder
$SessionVar.padded = lpad("12.3",20,"*");
```

Isyvar

The **lsyvar** function loads the Operator station defined system variables. Normally the variables are loaded from a display's OnEntry action.

Syntax **void** *lsyvar*(**string-expr** <DisplayName>)

| Apply to | Actions |
|----------|---------|
|----------|---------|

| | |
|--------------|------|
| Return value | None |
|--------------|------|

```
Example      lsyvar();
             $Trend_1.Object_1 = $SystemVar.Log1Obj;
             $Trend_1.Attrib_1 = $SystemVar.Log1Attrib;
```

Itrim

The **ltrim** function trims a *string* for all leading (left) *trimmer* characters.

Syntax **string** *ltrim*(**string-expr** <string> [, **string-expr** <trimmer>])

If *trimmer* is omitted a space ' ' is used for trimming.

Apply to Expressions.

| | |
|--------------|--------------------|
| Return value | The trimmed string |
|--------------|--------------------|

Example // Trim a string for all leading spaces.
 \$SessionVar.trimmed = ltrim(" Plat");

opendisplay

When the display access mode is MDI run, the **opendisplay** function leaves the current display open, and opens a second display in another window. When the display access mode is SDI run, this function is equivalent to **display** ([display](#) on page 303).

Syntax **void** *opendisplay*(**string-expr** <group-name>,
 string-expr <display-name>)

Apply to Push button Actions

Return value None

Example // When pushing the button "Menu" load the main menu display
 opendisplay("MAIN","MENU");

oscolor

The **oscolor** function converts the operator station string color specification to Display Services numeric color specification. Operator station colors from "A1" to "H16" are converted to Display Services colors 0-127. Other operator station colors are converted to a white color.

Syntax **integer** *oscolor*(**string-expr** <os-color>)

Apply to Expressions

Return value The Display Services color number (0-127).

Example lsyvar("Trend1");
 \$Trend_1.Color1 = oscolor(\$SystemVar.Color1);

osfont

The **osfont** function converts the operator station string font specification to Display Services numeric font specification. All operator station fonts from "FA10" to "FL60" are converted to Display Services fonts 0-71.

Syntax **integer** *osfont*(**string-expr** <os-font>)

| | |
|--------------|--|
| Apply to | Expressions |
| Return value | The Display Services font number (0-71). |
| Example | <pre>lsyvar("Trend1"); \$Trend_1.Font1 = oscolor("\$SystemVar.Font1");</pre> |

ostreatment

The **ostreatment** function converts the operator station integer treatment to a string.

| | |
|--------------|--|
| Syntax | string <i>ostreatment</i> (integer-expr <os-treatment>) |
| Apply to | Expressions |
| Return value | The converted value as a string. |
| Example | <pre>\$Trend_1.T1_Treatment = ostreatment(#SystemVar.Treatment1); update("Trend_1");</pre> |

print

The **print** function prints the current display on the printer of type *printer-type* connected to the printerqueue *queue-name*. The following printer types are supported:

- pjet - HP Paintjet (color) printer.
- la100 - Digital LA100 printer.
- ps - Postscript printer.

| | |
|--------------|--|
| Syntax | void <i>print</i> (string-expr <queue-name>, string-expr <printer-type>) |
| Apply to | Actions. |
| Return value | None |
| Example | <pre>// Print the current display print("XL300","pjet");</pre> |

rand

The **rand** function calculates a random integer number in the range of 0 to (*number*-1).

Syntax **integer** *rand*(**integer-expr** <number>)

Apply to Expressions

Return value The random number.

Example &Numeric_1.DataQuery = rand(100);
update("Numeric_1");

rearrange

This function rearranges a one-column matrix into a multi-column matrix. This function does not effectively handle uneven number of data points to a column. For instance, if you try to rearrange three rows of data (one column) to two columns, the result will be one row of two columns. The third piece of data will be lost.

Syntax *rearrange*(**string-expr** <element>, **integer-expr** <cols> [, **integer-expr** <major>])

element is the matrix to be re-arranged.

Cols specifies the number of columns.

Major specifies Row or Column Major:

1= Row Major (default). With Row Major, objects are listed in rows and their attributes are listed in columns.

0=Column Major. With Column Major, objects are listed in columns and their attributes are listed in rows.

Apply to Expressions

Return value None.

Example *rearrange*("Matrix_1",4);

replace

The **replace** function replaces a sub-string of a string with another string. If another string is not specified, the original string is removed.

| | |
|--------------|--|
| Syntax | string <i>replace</i> (string-expr <string>, string-expr <search-string> [, string-expr <replacement-string>]) |
| Description | <i>search-string</i> is the string to be replaced <i>replacement-string</i> is the new string. If the <i>replacement-string</i> is omitted the <i>search-string</i> is removed. |
| Apply to | Expressions. |
| Return value | The replaced string |
| Example | // Replace the word "Line_1" with "Line_2" \$SessionVar.lines = replace(\$SessionVar.lines,"Line_1","Line_2"); |

return

The **return** function returns a value from a user defined function.

| | |
|--------------|--|
| Syntax | <i>return</i> <returnexpr> The <i>returnexpr</i> can be any type. |
| Apply to | Functions |
| Return value | <i>returnexpr</i> |
| Example | if (&SessionVar.Status) then return 0; else .. |

rpadd

The **rpadd** function pads the *string* to the specified *number* of characters.

| | |
|--------|---|
| Syntax | string <i>rpadd</i> (string-expr <string>, |
|--------|---|

integer-expr <number> [,
string-expr <padder>)]

The *padder* character is used for padding. If *padder* is omitted, a space ' ' is used for the padding.

| | |
|--------------|--|
| Apply to | Expressions. |
| Return value | The padded string |
| Example | // Pad a string to 20 in length. Use the '*' as padder \$SessionVar.padded = rpad("12.3",20,"*"); |

rtrim

| | |
|--------------|--|
| Function | The rtrim function trims a string to the right (trailing characters). |
| Syntax | string <i>rtrim</i> (string-expr <string> [, string-expr <trimmer>)] <i>trimmer</i> is the character string to trim. If <i>trimmer</i> is omitted, a space ' ' is used for trimming. |
| Apply to | Expressions. |
| Return value | The trimmed string |
| Example | // Trim a string for all trailing spaces. \$SessionVar.trimmed = rtrim("Plat"); |

setfocus

The **setfocus** function sets the input focus to the specified edit-field. This function is PC-specific.

| | |
|--------------|--|
| Syntax | <i>setfocus</i> (string-expr <number>) |
| Apply to | Expressions |
| Return value | None. |
| Example | setfocus(edit_1); |

| | |
|--------------|--|
| Apply to | Dataquery |
| Return value | The data requested in the SQL statement. Data type as defined. |
| Example | <pre>// Get all productions defined on "Line_1" // limit the number of productions returned to 25 \$\$sql("SELECT PRODUCTION FROM SVF_PRODUCTIONS WHERE LINE_NO='Line_1'",25);</pre> |

sqrt

The **sqrt** function calculates the positive square root of a *number* into a float.

| | |
|--------------|---|
| Syntax | float <i>sqrt</i> (float-expr <number>) |
| Apply to | Expressions |
| Return value | Square root of <i>number</i> |
| Example | <pre>&Numeric_1.DataQuery = 12 + sqrt(23); update("Numeric_1");</pre> |

substr

The **substr** function extracts a substring from a string.

| | |
|--------------|---|
| Syntax | string <i>substr</i> (string-expr <string>, integer-expr <start-position>, integer-expr <length>) |
| Description | <i>string</i> is the return string, starting at <i>start-position</i> and of length <i>length</i> . |
| Apply to | Expressions |
| Return value | The specified substring. |
| Example | <pre>// Get the first two letters of the current user // Note the "I" construction to produce a correct dataquery syntax \$Text_1.DataQuery = "I"+substr(getenv("USER"),1,2) + "I"; update("Text_1");</pre> |

system

The **system** function executes a system (shell) command. Depending on the value of the *on-host* parameter, the command is executed on the client (*on-host* = 0) or on the host (*on-host* = 1). The default for *on-host* is 0.

If the **system** function is run at the client, it must be preceded by a test of the operating system (see [ishost](#)) to be sure the correct system call in configurations where Display Services runs on multiple platforms (UNIX, DOS/WINDOWS).



The system function is asynchronous. If it is used to issue a call on a remote system, the command will not wait for the results to be completed.

Syntax **void** *system*([**integer** <channel>],[
 string-expr <command>[, **integer-expr** <on-host>])

<channel> is an optional parameter for referencing a specific data provider when there are more than one data provider of the same type. If <channel> is not specified, the function will access the default data provider which is CHANNEL 0.

Apply to Actions

Return value None

Example if (ishost("UNIX")) then system("hpterm&");
 endif;
 ..

// When pushing the button "Clock" start a xclock session
 system("xclock&");



The System function requires the user preference for system access to be set to true. Refer to the *System 800xA Information Management Configuration (3BUF001092*)* for details on configuring user preferences.

tan

Function The **tan** function calculates the tangent of a float. The angle is given in radians.

Syntax **float** *tan*(**float-expr** <number>)

Apply to Expressions

Return value The tangent value.

Example &Numeric_1.DataQuery = 12 + tan(0.4);
 update("Numeric_1");

time

This function performs one of the following functions, depending on the syntax:

- [Return the Number of Seconds Elapsed Since 1-1-1970](#)
- [Format the Current Time Into a String](#)
- [Format a Specified Time Sec \(Elapsed Since 1-1-1970\) Into a String](#)
- [Return Textual Representation of Specified Time Sec \(Elapsed Since 1-1-1970\)](#)
- [Convert Date/Time String Into Seconds Elapsed Since 1-1-1970](#)



The application for converting the date/time string into seconds elapsed since 1-1-1970 is not supported by the HP-UX server platform. You can only use this function on the PC-based server.

Return the Number of Seconds Elapsed Since 1-1-1970

Syntax **integer** *time*()

Apply to Expressions

Return value Number of seconds Elapsed since 1-1-1970

Example #Numeric_1.DataQuery = time();
 update("Numeric_1");

Format the Current Time Into a String

Syntax **string** *time*(**string-expr** <format>)

 See [Table 64](#) for a listing of format specifiers.

Apply to Expressions

Return value None

Example \$Text_1.DataQuery = "I"+time("%d-%m-%y")+ "I";
 update("Text_1");

Format a Specified Time Sec (Elapsed Since 1-1-1970) Into a String

Syntax **string** time(**string-expr** <format>, **integer** <sec>)

See [Table 64](#) for a listing of format specifiers.

Apply to Expressions

Return value None

Example #Numeric_1.DataQuery = time(%, 60);
 \$Text_1.DataQuery = "I"+time("%d-%m-%y")+ "I";
 update("Numeric_1");
 update("Text_1");

Return Textual Representation of Specified Time Sec (Elapsed Since 1-1-1970)

Syntax **string** time(**integer** <sec>)

Apply to Expressions

Return value None

Example \$Text_1.DataQuery = time(60);
 update("Text_1");

Convert Date/Time String Into Seconds Elapsed Since 1-1-1970

This value is required when the data function is used to add or modify a log value as described in [data](#) on page 295.

Syntax **integer** time(**string-expr** <format>, **string** <date/time>)

See [Table 64](#) for a listing of format specifiers.



For this version of the time function, only the following specifier formats can be used: %d , %H , %m, %S, %y, %Y, %%.

Apply to Expressions

Return value None

Example

#Numeric_1.DataQuery = time("%d-%m-%y %H:%M:%S",
"11-08-2000 14:48:30");
update("Numeric_1");

Table 64. Format Specifiers

| Specifier | Description |
|-----------|--|
| %a | Locale’s abbreviated weekday name. |
| %A | Locale’s full weekday name. |
| %b | Locale’s abbreviated month name. |
| %B | Locale’s full month name. |
| %c | Locale’s appropriate date and time representation. |
| %d | Day of the month as a decimal number [01,31]. |
| %E | Locale’s combined Emperor/Era name and year. |
| %H | Hour (24-hour clock) as a decimal number [00,23]. |
| %I | Hour (12-hour clock) as a decimal number [01,12]. |
| %j | Day of the year as a decimal number [001,366]. |
| %m | Month as a decimal number [01,12]. |
| %M | Minute as a decimal number [00,59]. |
| %n | New-line character. |
| %N | Locale’s Emperor/Era name. |
| %o | Locale’s Emperor/Era year. |
| %p | Locale’s equivalent of either AM or PM. |
| %S | Second as a decimal number [00,61]. |
| %t | Tab character. |
| %U | Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0. |

Table 64. Format Specifiers

| Specifier | Description |
|-----------|--|
| %w | Weekday as a decimal number [0(Sunday),6]. |
| %W | Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0. |
| %x | Locale's appropriate date representation. |
| %X | Locale's appropriate time representation. |
| %y | Year without century as a decimal number [00,99]. |
| %Y | Year with century as a decimal number. |
| %Z | Time zone name (or by no characters if no time zone exists). |
| %% | The character % |

trace

The **trace** function prints the *trace-value* on the terminal window where Display Services was started. If the *trace-value* is omitted a carriage return is written.

Syntax **void** *trace*([**expr** <trace-value>])
Trace-value can be any string, integer or float.

Apply to Actions

Return value None

Example a text element
 `trace($Text_1.Value);`
 `trace();`

update

The **update** function forces updating of a display element with the specified *element-name*. The function is normally called after a change of an element's property in runtime.

| | |
|--------------|--|
| Syntax | void <i>update</i> (string-expr <element-name>) |
| Apply to | Actions |
| Return value | None |
| Example | // Change the shape of a shape element to type BOX \$Shape_1.Shape = "Box"; <i>update</i> ("Shape_1"); |

upper

The **upper** function returns the *string* with all characters converted to uppercase.

| | |
|--------------|---|
| Syntax | string <i>upper</i> (string-expr <string>) |
| Apply to | Expressions |
| Return value | The converted string. |
| Example | \$SessionVar.upc = <i>upper</i> ("Abdominal"); |

```

sql("insert into Proceskontrollkort(
Maalenr,
Mask_nr,
Pro_nr,
Antal,
Kant_max,
Kant_min,
Man,
Vis,
Operator,
Tekst
)
values(
substr('"+$SessionVar.MaaleNr+"',1,4),
substr('"+$SessionVar.Mask_nr+"',1,40),
substr('"+$SessionVar.Pro_nr+"',1,40),
substr('"+$SessionVar.Antal+"',1,40),
substr('"+$SessionVar.Kant_max+"',1,40),
substr('"+$SessionVar.Kant_min+"',1,40),
substr('"+$SessionVar.Man+"',1,60),
substr('"+$SessionVar.Vis+"',1,40),
substr('"+$SessionVar.Operator+"',1,40),
',' )
");

$SessionVar.MaaleNr=$sql("select MAX(Maalenr)+1 from proceskontrollkort");
update(MaaleNr);
$SessionVar.Mask_nr="";
update(Mask_nr);
$SessionVar.Pro_nr="";
update(Pro_nr);
$SessionVar.Antal="";
update(Antal);
$SessionVar.Kant_max="";
update(Kant_max);
$SessionVar.Kant_min="";
update(Kant_min);
$SessionVar.Man="";
update(Man);
$SessionVar.Vis="";
update(Vis);
$SessionVar.Operator="";
update(Operator);
execute("Mask_nr_1st","ReQuery");
execute("Pro_nr_1st","ReQuery");
execute("Operator_1st","ReQuery");
execute("Antal_1st","ReQuery");
execute("Kant_min_1st","ReQuery");
execute("Kant_max_1st","ReQuery");
execute("Man_1st","ReQuery");
execute("Vis_1st","ReQuery");
$sql("select to_char(sysdate,'hh24:mi') from dual")

```

Figure 175. Example Scripts

Appendix A A Demonstration of Display Services

Introduction

This demonstration shows you how to browse demonstration displays provided with Display Services. You can look at display elements, and their property definitions to learn how to build similar displays for your own application. You can even use scripts in these example displays, and adapt them to your application, sometimes by simply changing the tag reference.

The tour does not show all the demo displays, nor all of the different display elements. After finishing this quick tour, you should be able to browse the displays.



This tour demonstrates display building features on a PC client. The ADO-DEMO data provider must be running. How to manage data providers can be found in *System 800xA Information Management Configuration (3BUF001092*)*.

A Quick Tour via the PC Client

Start the tour by logging into the PC client as a valid user in the Build mode. The Build mode lets you view displays in both the Run mode and Build mode. See [Startup](#) on page 21 for details.

Access to all displays is via the Object Browser, [Figure 176](#). The starting point for this tour is the Demostart display object in the DisplayDemo group. To navigate to and run this display, first click the (+) button next to display server object. This shows the display groups residing on this display server. In [Figure 176](#), the display server is tar237.

Next, click the (+) button next to the folder for DisplayDemo group. This is the group where the Demostart display object is located.

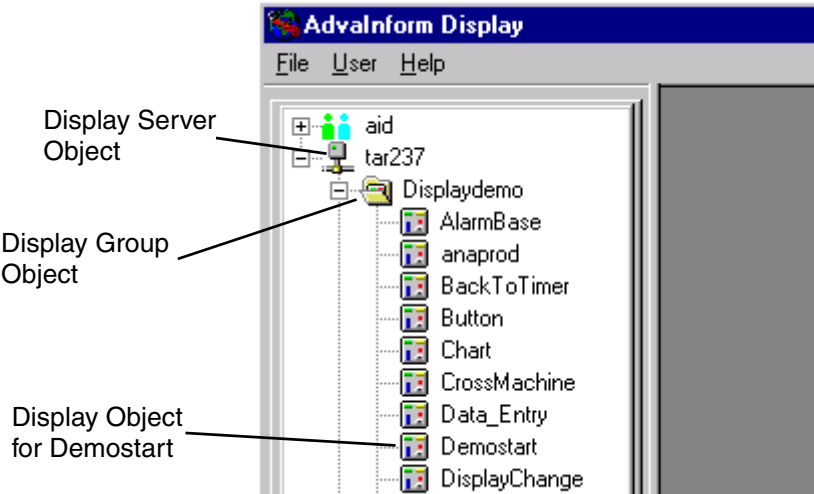


Figure 176. Starting the Tour

To open the Demostart display in run mode, click on the Demostart object to select it, then right-click and choose **Run** from the context menu, [Figure 177](#).

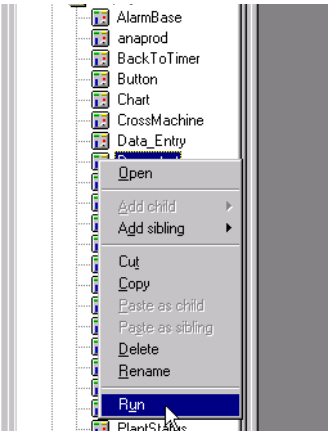


Figure 177. Starting a Display in Run Mode

Demostart is the home display for all demo displays, [Figure 178](#).

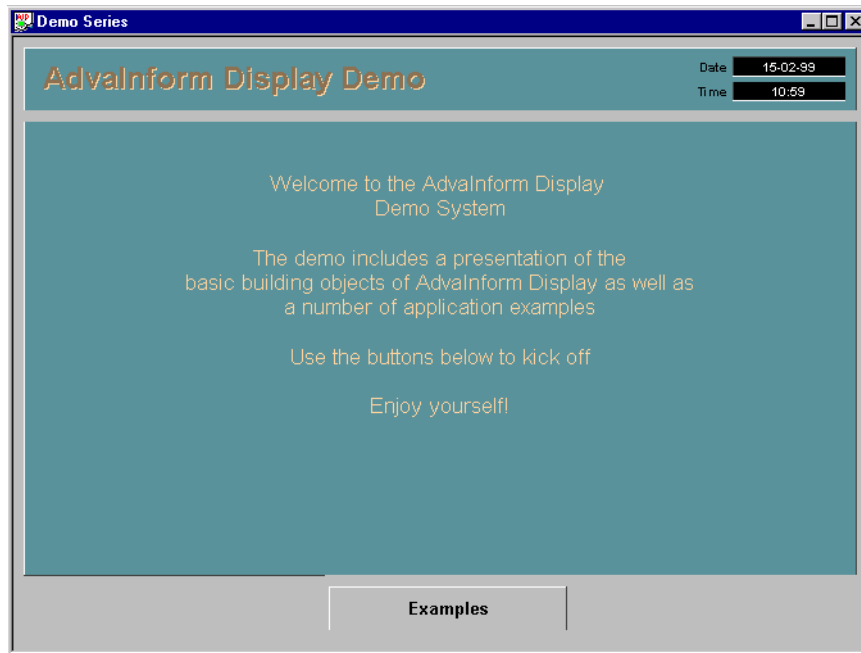


Figure 178. Demostart

Click **Examples**. This opens the Overview display, [Figure 179](#). This display provides buttons for launching the various example displays. There are two sets of demo displays. One set shows examples of the various basic objects (display elements). The other provides examples of complete displays for common applications.

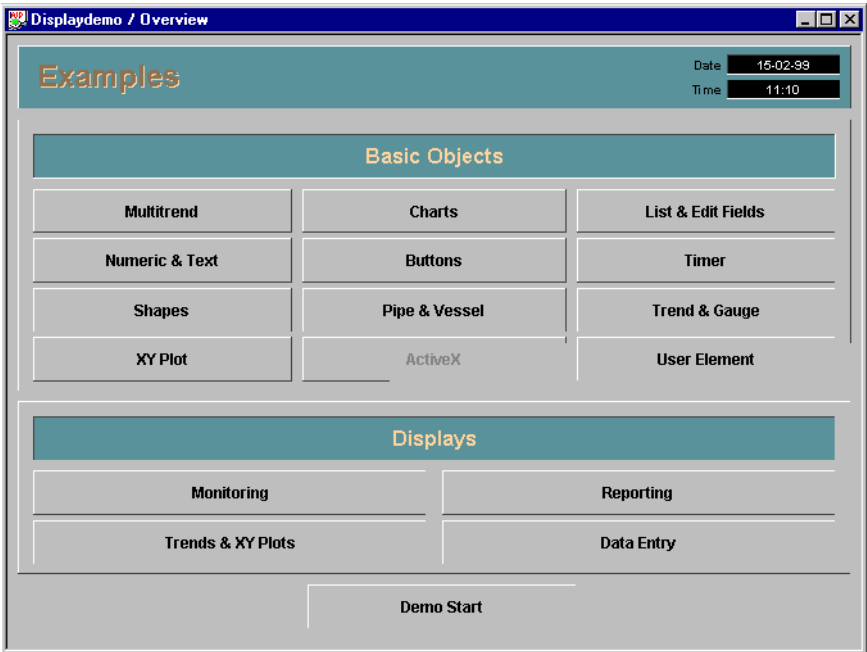


Figure 179. Overview Display

You can return to the Demostart display from this overview by clicking the **Demo Start** button.

Basic Objects

To go to the demo display for a particular display element, simply click on the corresponding button under Basic Objects. For a quick demonstration, click on the **Lists & Edit Fields** button. The List & Edit Fields display shows examples of List, Combobox, and Edit display elements, [Figure 180](#).

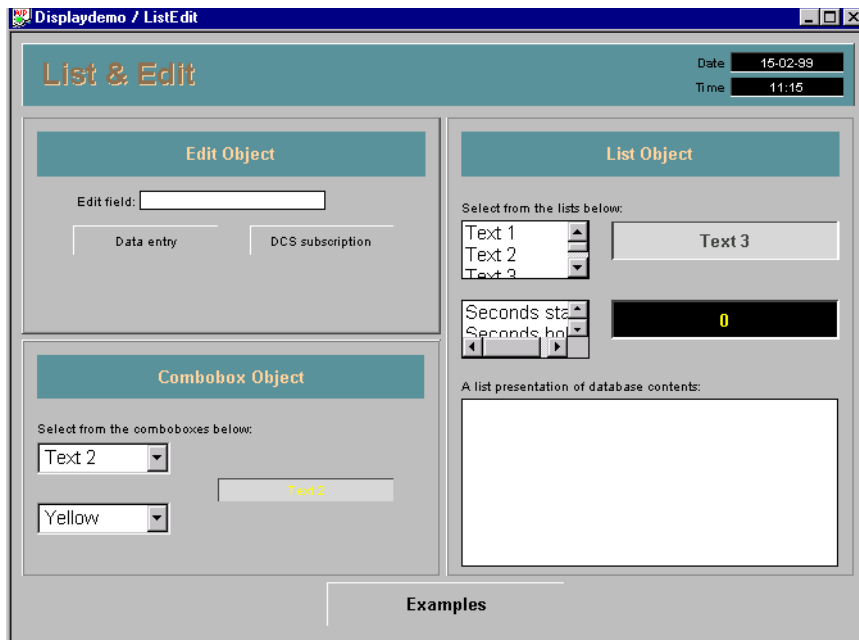


Figure 180. Run Mode View of Lists & Edit Fields Display



To return to the Overview display from this or any other example display, simply click the **Examples** button.

You are now looking at the Run mode view of the List & Edit Fields display. To see how a particular display element is defined, open the Build mode view. To do this, using the Object Browser, navigate to the List & Edit display object in the DisplayDemo group, and double-click.

Now you have two views of the same display open, [Figure 180](#).

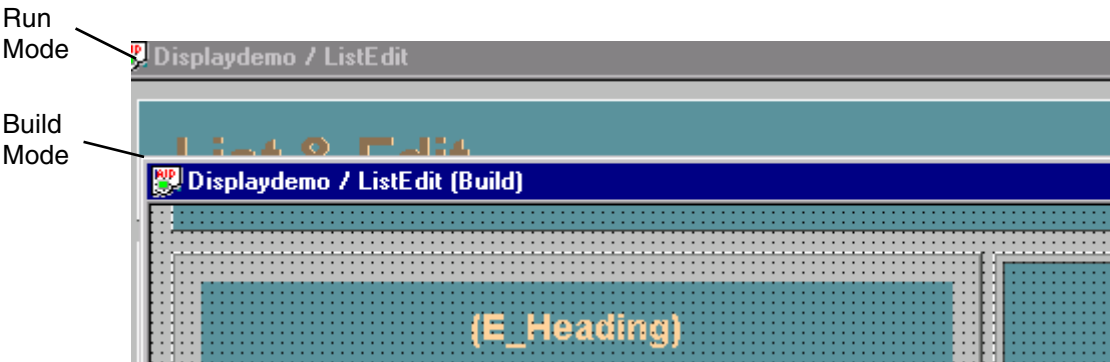


Figure 181. Build Mode View of EDIT & LIST Display

To look at the properties for a display element, first click on a display element to select it, then right-click and choose **Properties** from the context menu.



You may have to ungroup parts of the display to select a specific display element, if display elements are grouped. To ungroup, select the portion of the display that you want to ungroup, and then choose **Change > Arrange > Ungroup** from the main menu bar.

For example, look at the properties for the Combobox_2 display element in the lower left corner of the display, [Figure 182](#).

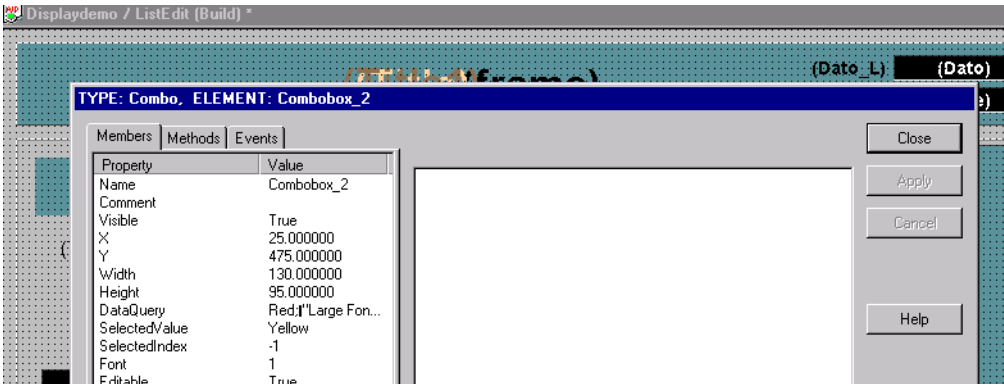


Figure 182. Properties Dialog for Selected Display Element

To see how a particular property is defined, click the appropriate tab where the attribute is listed, and then click on the property to select it.

For instance, ItemSelected is an Event-type property that specifies the action to occur when you select an item in a list. To look at the ItemSelected property for this Combobox display element, first click the **Events** tab, and then click ItemSelected. The action is defined in a script, [Figure 183](#).

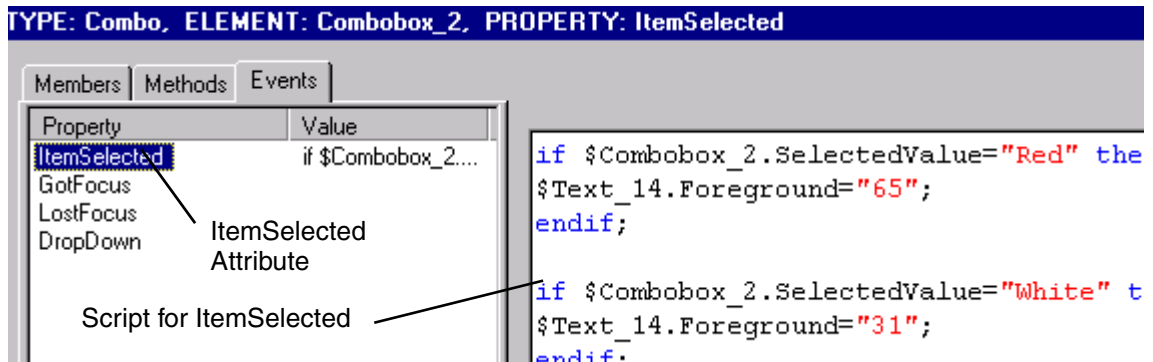


Figure 183. Taking a Closer Look at a Display Element Property

The script specifies ways to manipulate the text in Text display element no. 14, depending on the item you select in Combobox_2. Test the operation of these elements by returning to the Run mode view of the Edit and List display.

Select an item in Combobox_1, and watch the contents of Text_14 change accordingly. For instance, if you select Green from the list, the text in Text_14 is displayed in green, [Figure 184](#).

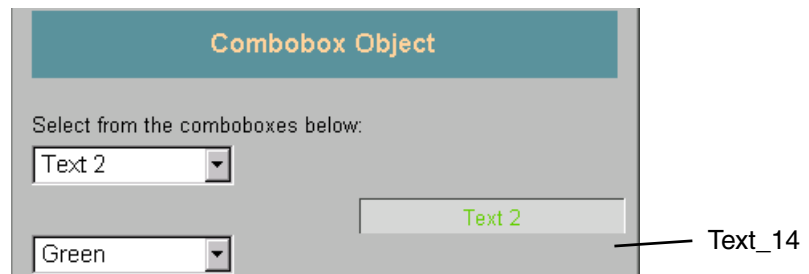


Figure 184. Testing the Display Element Operation

To close the Build view for this display, click the Windows close button (X) in the top right corner. Click the **Examples** button to return to the Overview display and select other display element examples.

Example Displays

Click any one of the Display buttons (Monitoring, Data Entry, Reporting, Trends & XY Plots) to look at examples of complete displays. For example, clicking the **Monitoring** button, displays the Process Status display, [Figure 185](#).

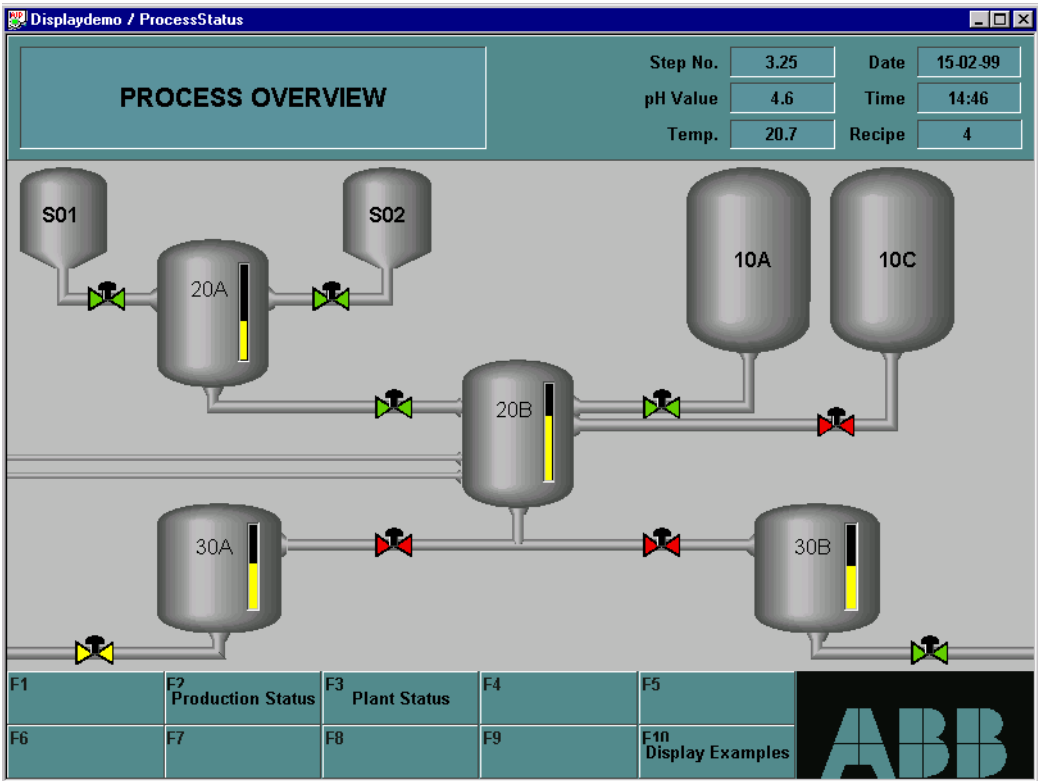


Figure 185. ABBButton Display Element for Navigating Example Displays

Simply click on the appropriate button to go to the display you want to see. The **Display Examples** button takes you back to the Overview display, [Figure 179](#).

Appendix B More Tutorials

Tutorial 1, Hello World!

As with any other new language the most basic exercise is to create a script that writes the text "Hello world!". This exercise demonstrates how to write scripts to produce the text in a text display element. You should already be able to create a display with the appropriate display elements. The display element properties required for this tutorial are given in the course of this tutorial. To do this exercise, follow these instructions:

1. Create a display with a Text element and a Button element.
2. Define the properties of the Text element as follows:

| | |
|--------|--------|
| Name | Text_1 |
| X | 25 |
| Y | 25 |
| Width | 500 |
| Height | 80 |

3. Define the properties of the Button element as follows:

| | |
|---------|----------|
| Name | Button_1 |
| X | 25 |
| Y | 100 |
| Width | 200 |
| Height | 40 |
| Caption | "Hello" |

4. Define the Action property of the Button element using the following script:

```
// First set the properties in "Text_1"  
$Text_1.DataQuery = "!Hello World!!";  
&Text_1.Font = 5;  
&Text_1.Foreground = 66;  
&Text_1.Background = 0;
```

```
// Then update the "Text_1"
update("Text_1");
```

The action is defined in a script which may be entered directly in the Properties window as shown in [Figure 186](#).



The pipe character (|) and extra set of quotation marks in the line for Text_1.DataQuery above are means for embedding quotation marks within the text string. The actual DataQuery for Text_1 is "Hello World!", rather than Hello World!. This same method can be used to embed spaces in a text string. The pipe character means to take the next character (in this case the double quotation mark) literally rather than as a delimiter.

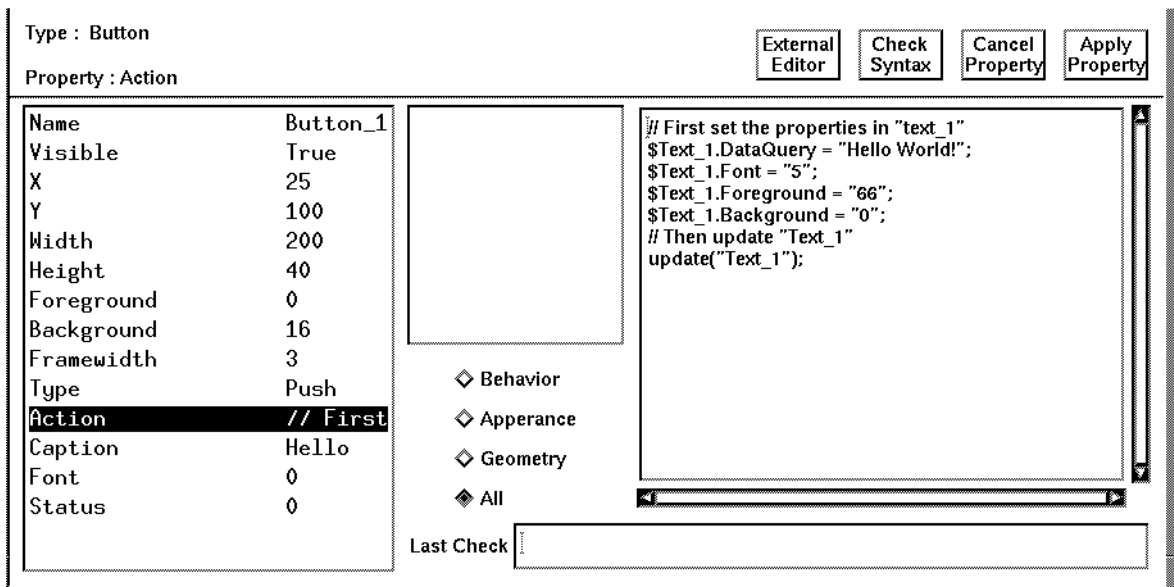


Figure 186. Defining Action in a Script in Properties Window

- Choose **File > Save** to save the display. To run the display select the display icon in the Object Browser, right click and choose **Run** from the context menu.
- Push the Button labeled Caption. The text field should now display the text: Hello World!.

The script in step 4 should be self-explanatory. If you need a little help to understand what each statement does, read the following explanation.

The statements that are preceded by the `//` characters are comments. `//` indicates that what ever follows on this line is non-executable, and for information only. If a comment requires more than one line, each line must be preceded by the comment delineator (`//`). Refer to [Section 6, Display Scripting](#).

The four statements that follow the first comment define certain properties of the `Text_1` display element. For instance `$Text_1.DataQuery = "Hello World!"` defines the data query property as the text string "Hello World". The dollar sign (\$) indicates that what follows is a single text string variable. Multitext string variables are preceded by `$$`. Integers are preceded by an ampersand (&), and floating point values are preceded by a pound sign (#). Refer to [Section 6, Display Scripting](#) for further information regarding data types.

Finally, the statement following the second comment updates the `Text_1` display element properties with the values indicated in the lines above. Further details regarding update and other functions are provided in [Section 6, Display Scripting](#).

Tutorial 2, An Object Display

This tutorial creates a display to show the Value and Description of an Analog Input Object (AI). To do this exercise, follow these instructions:

1. Create a new display with a Numeric, Text and Edit Element.
Name them "Numeric_1", "Text_1" and "Edit_1" respectively. Place them as you like, and choose the color and font as you like.
2. Change the "DataQuery" property of "Numeric_1" to:

```
#dcssub($Edit_1.Value, "AI", "VALUE", dcs_3s)
```

3. Change the "DataQuery" property of "Text_1" to:

```
$dcsub($Edit_1.Value, "AI", "DESCRIPTION", dcs_demand)
```

4. Change the "Activate" property of "Edit_1" to:

```
execute("Numeric_1", "ReQuery");  
execute("Text_1", "ReQuery");
```

Save the display and run it. Try to type-in a known reachable AI object and press return. The value and description of the specified object appear in the two text fields

Appendix C Log Manager Trend Display Guidelines

Introduction

The Log Manager Trend display, [Figure 187](#), is one of three trend displays provided with the demo displays described in [Appendix A, A Demonstration of Display Services](#). The data source for this trend display is the log manager which supports data retrieval from History logs and TTD logs.

The Log Manager Trend display includes the following features: Trace Configuration, Time Scope adjustment functions, Ruler functions, Zoom functions, Time Offset functions, Filter functions.

How to Proceed

To access the Log Manager Trend Display, see [Accessing the Log Manager Trend Display](#) on page 340.

To specify which logs to display see [Configuration](#) on page 341.

You can customize certain display properties. See [Customizing the Display](#) on page 343.

Guidelines for using the display are provided in [Operation](#) on page 344.

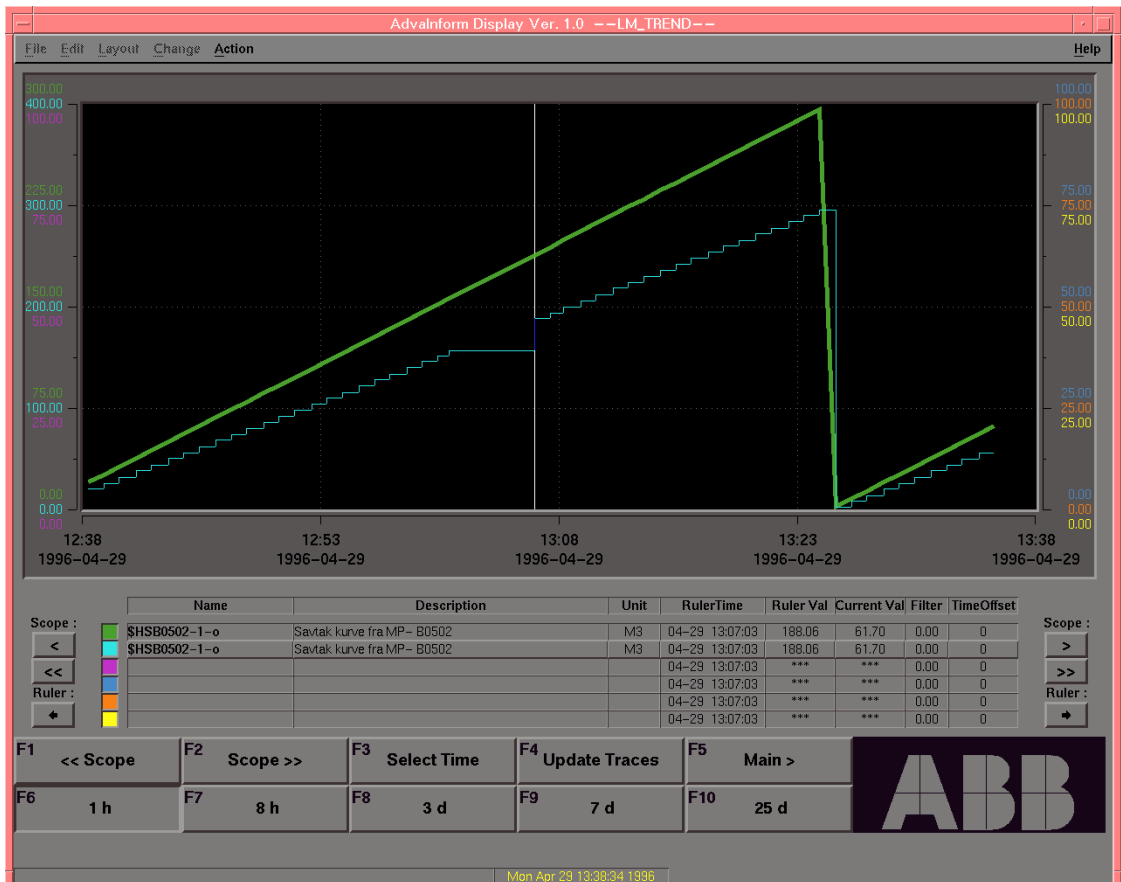


Figure 187. LM_TREND Display

Accessing the Log Manager Trend Display

The LogManagerTrend Display belongs to the Displaydemo group. To open this display, find the display under Demodisplay in the [Object Browser](#), then right click and choose **Run** from the context menu, [Figure 188](#).

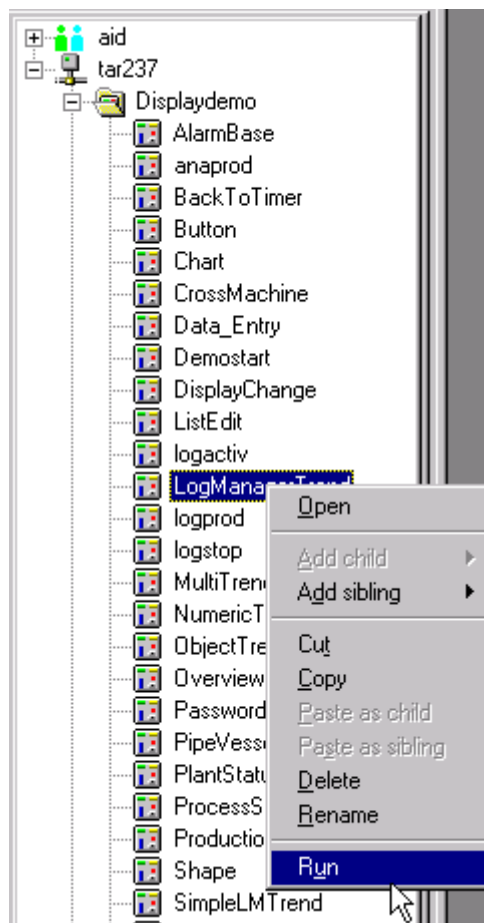


Figure 188. Opening the LogManagerTrend Display in Run Mode

Configuration

This display requires very little configuration to make it operational. Simply specify the logs to display. You can display up to six logs at a time. To specify the logs:

1. Using the F1-F10 button, [Figure 189](#), click the following buttons in this sequence:

| | | | | | | | | | |
|----|----------|----|----------|----|-------------|----|---------------|-----|--------|
| F1 | << Scope | F2 | Scope >> | F3 | Select Time | F4 | Update Traces | F5 | Main > |
| F6 | 1 h | F7 | 8 h | F8 | 3 d | F9 | 7 d | F10 | 25 d |

Figure 189. Default Menu Bar

- a. First click **Main**. This displays a new menu bar, [Figure 190](#):

| | | | | | | | | | |
|----|---------|----|--|----|---------------------|----|-----------|-----|------------|
| F1 | Zoom... | F2 | | F3 | Scope Adjustment... | F4 | Traces... | F5 | < Main |
| F6 | | F7 | | F8 | | F9 | | F10 | Trend Menu |

Figure 190. New Menu Bar

- b. Next click on **Traces**. This displays another new menu bar, [Figure 191](#).

| | | | | | | | | | |
|----|-----------|----|------------|----|--------------|----|------------------|-----|------------|
| F1 | Y max/min | F2 | Hide Trace | F3 | Time Offset | F4 | Configuration... | F5 | Next Trace |
| F6 | | F7 | Show Trace | F8 | Filter Param | F9 | | F10 | Main |

Figure 191. Another New Menu Bar

2. Then click **Configuration**. This displays an edit field below the menu bar.
3. In this edit field, enter the name of a log, [Figure 192](#). The format can be:

logname (for example: B0502) OR
\$HS<*logname*>-1-o (for example: \$HSB0502-1-o)

| | | Name | Description | Unit | RulerTime | RulerVal | Current Val | Filter | TimeOffse |
|--------|--|--|-------------|------|-----------|----------|-------------|--------|-----------|
| Scope: | | B0502 | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| Ruler: | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | | | | xxx | xxx | xxx | 0.00 | 0 |
| | | Source Object = <input type="text" value="B0502"/> | | | | | | | |

Figure 192. Entering the Log Name

4. Press ENTER. This displays the requested trend, and the name is displayed in the Trend Info area.

If a trend is not displayed when requested, check the following:

- Check that the button next to the Trend Info area is set to show (not hide) the trend.
- Make sure the color of the trend is visible on the trend background. The color of a trend is shown in the button next to Trend Info area.
- Make sure the range of the logged values is within the Y-scale range of the trend.
The default Y-scale is 0 to 100. To change the Y-scale. See [Trace](#) on page 351.
- Make sure the logname you entered is spelled correctly.



When you save the display, all settings are saved. This includes the selected trend, the time scope, the currently active Menu Bar and so on. If you want to preserve the original demo display, save the display under a new name using the **Save As** menu item.

Customizing the Display

As an option, you can change one or more trend display element properties to customize your Log Manager Trend display. Some common properties are listed below:

- Background color of the trend area - Change the property **TrendBackground** in the Trend element.
- Color of each trend - Change the property **Tn_Color** in the Trend element.
- Max. and min. Y-scale values - Change the property **Tn_Scalemax** and **Tn_Scalemin** in the Trend element.
- Fontsize of Y-scale - Change the property **Tn_Font** in the Trend element.
- Format of the Y-scale label - Change the property **Tn_Valueformat** in the Trend element.

To open the LogManagerTrend Display in Build mode, you must start the PC client in Build mode. See [Startup](#) on page 21. Then find the display under Demodisplay in the [Object Browser](#), and either double-click, or right click and choose **Run** from the context menu.

For instructions on how to modify trend display element properties, refer to [Trend](#) on page 170.

Operation

The Log Manager Trend Display consists of these main parts - the Trace Area, Trend Info Area, and Menu Bar - as shown in [Figure 193](#).

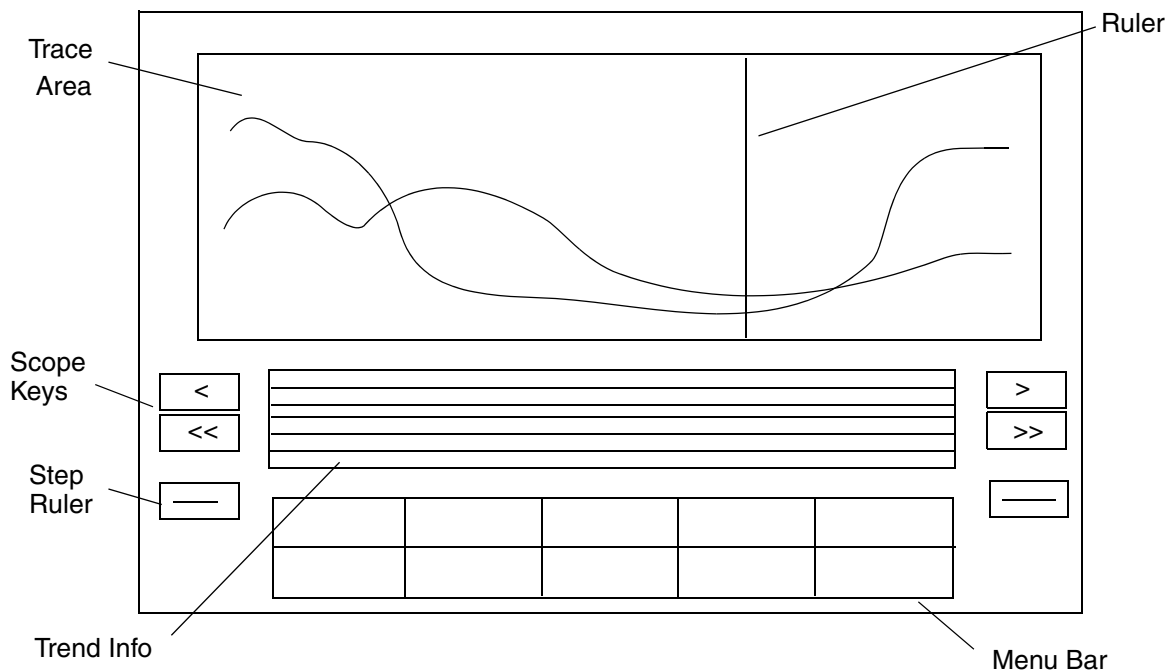


Figure 193. LM_TREND Display Main Parts

For operating guidelines, see:

- [Trace Area](#) on page 345

- [Ruler](#) on page 345
- [Trend Info](#) on page 345
- [Scope Keys](#) on page 347
- [Ruler Step Keys](#) on page 347
- [Menu Bar](#) on page 347

Trace Area

The trace area displays up to six trends simultaneously. The color of each trend can be selected from a color palette. The representation of each trend can be either *Linear* or *SampleHold* as shown in [Figure 194](#).

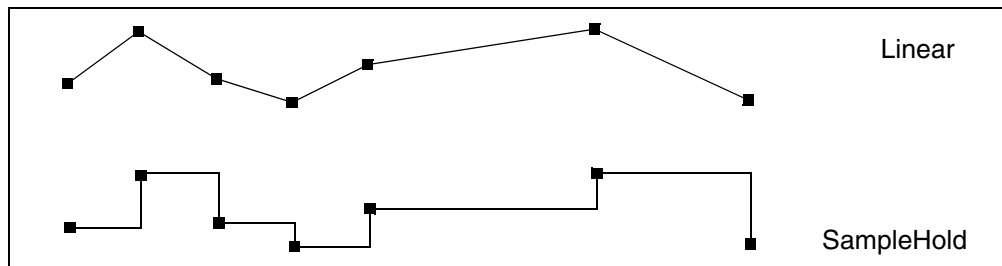


Figure 194. Examples: Linear & SampleHold

Linear trends are drawn point-to-point. In SampleHold trends, the value is drawn as a horizontal line until a new value is reached.

Ruler

When you click the mouse inside the Trace Area, a Ruler (the vertical line) is displayed. The Ruler is placed at the nearest value of the selected trend (the selected trend is indicated in the Trend Info area). The time and value of all trends having data at the ruler time, is also displayed in the Trend Info area.

Trend Info

The Trend Info area contains information on all displayed trends, such as name, description, unit and values, [Figure 195](#). The selected trend is highlighted. To select

another trend, simply click somewhere on the line corresponding to the desired trend.

Show/Hide

Selected trend

| | Name | Description | Unit | RulerTime | Ruler Val | Current Val | Filter | TimeOffset |
|--|---------------|----------------------------|------|----------------|-----------|-------------|--------|------------|
| | \$HSB0502-1-o | Savtak kurve fra MP- B0502 | M3 | 04-29 12:48:57 | 79.52 | 271.65 | 0.00 | 0 |
| | | | | 04-29 12:48:57 | *** | *** | 0.00 | 0 |
| | | | | 04-29 12:48:57 | *** | *** | 0.00 | 0 |
| | | | | 04-29 12:48:57 | *** | *** | 0.00 | 0 |
| | | | | 04-29 12:48:57 | *** | *** | 0.00 | 0 |
| | | | | 04-29 12:48:57 | *** | *** | 0.00 | 0 |

Figure 195. Trend Info Area

The columns in the Trend Info area are described below:

- Name - Name of log to display. Retrieved from the Trend element.
- Description - Description of log. Retrieved from LogManager.
- Unit - Unit of the log. Retrieved from LogManager.
- RulerTime - Time at the position where the ruler is placed. If no ruler is placed the field contains ‘***’. Updated from the Trend element.
- RulerValue - Value at the position where the ruler is placed. If no ruler is placed or no data exists at that point, the field contains ‘***’. Updated from the Trend element.
- CurrentValue - Current value of the log, independent of the actual time scope. Updated from the Trend element.
- Filter - Factor with which the trend is filtered. When a filter factor is applied, the field is highlighted white. See [Trace](#) on page 351 for more information on the Filter function.
- TimeOffset - Time the trend is pushed forward or backward in time compared to the other trends. When an offset is applied, the field is highlighted white. See Chapter for more information on the TimeOffset function.
- Show/Hide - These buttons (left side of the Trend Info area) are used to show or hide the corresponding trend. If the button is pushed in the trend is shown. If the button is popped out the trend is hidden and the text in the Trend Info line is dimmed.

Scope Keys

Each side of the Trend Info area has two buttons for scope adjustment, [Figure 196](#).



Figure 196. Scope Keys

Clicking one of these buttons moves the scope the indicated percentage of the full scope in the indicated direction. For example, if the time scope is one hour, the 20% button will move the scope 12 minutes and the 80% button will move the scope 48 minutes. This function is also available in the Menu Bar.

Ruler Step Keys

The Ruler Step keys, [Figure 197](#), are located below the Scope keys. After placing the ruler in the trend area, it is possible to move the ruler to the nearest value for the selected trend, either to the left or to the right.



Figure 197. Ruler Step Keys

This function is also available in the Menu bar.

Menu Bar

The Menu Bar is located near the bottom of the display. It has six levels, [Figure 198](#).

- **Main1** - Contains functions for selecting scope and updating traces. Main1 is the default level when you enter the display.
- **Main2** - Used for selecting other menus.
- **Zoom** - Functions for zooming, either controlled by mouse or by entering data from keyboard.

- [Scope Adjustment](#) - Functions for changing scope and moving the ruler.
- [Trace](#) - Functions for changing parameters for each trend, such as Y-scale max/min values, filter values, time offset, and so on.
- [Configuration](#) - Used when entering the log to display.

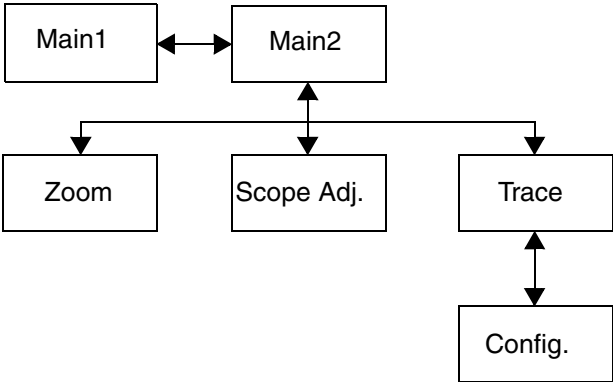


Figure 198. Menu Levels

Main1

The Main1 menu is shown in [Figure 199](#).

| | | | | |
|----------------|----------------|-------------------|---------------------|--------------|
| F1 << Scope | F2 Scope >> | F3 Select Time | F4 Update Traces | F5 Main > |
| F6 1 h | F7 8 h | F8 3 d | F9 7 d | F10 25 d |

Figure 199. Main1 Menu

This menu provides the following functions:

- F1-F2: Scope << & >> - Moves the current scope 80% to the left or to the right (just like the Scope Keys next to the Trend Info area).
- F3: Select Time - Is used for manually entering the desired Endtime of the current scope. First enter the time, press <Enter> and then enter the date you

wish to see. The current scope is maintained, and the Endtime is set to the value you entered.

- F4: Update Traces - Sets the Endtime to current time and maintains the scope and unlocks the Trend Area if it was locked as a result of zooming.
- F5: Main > - Switches to Main2 menu.
- F6-F10: 1h, 8h, 3d, 7d, 25d - Scope selection buttons, used for selecting scope using the current time as Endtime.

Main2

The Main2 menu buttons, [Figure 200](#), are used to switch to other menus.

| | | | | |
|---------------|----|-----------------------|-----------------|--------------|
| F1 Zoom... | F2 | F3 Scope Adjust... | F4 Traces... | F5 < Main |
| F6 | F7 | F8 | F9 | F10 Menu |

Figure 200. Main2 Menu

The functions provided by Main2 are described below:

- F1: Zoom... - Switches to the Zoom menu.
- F3: Scope Adjustment... - Switches to the Scope Adjustment menu.
- F4: Traces... - Switches to the Traces menu.
- F5: < Main - Switches to the Main1 menu.
- F10: Menu - Changes display to return to the Display demo.

Zoom

The Zoom menu contains functions for modifying the visible timespan, [Figure 201](#). When you change the Endtime to anything but the actual time, the Trend Area is locked. This is indicated by a yellow border around the Trend Area. This means that the trends not will be dynamically updated when data are received from the Log Manager.

| | | | | |
|-----------------------|----------------|-----------------|----|-------------|
| F1 Time & Duration | F2 | F3 | F4 | F5 |
| F6 Zoom In | F7 Zoom Out | F8 Area Zoom | F9 | F10 Main |

Figure 201. Zoom Menu

The functions provided by the Zoom menu are described below:

- F1: Time & Duration - Is used for selecting a user defined scope. Enter the Endtime, Enddate and Scope. The Scope can be entered in various formats, for example:
3600 (secs. = one hour), 5:00 (min:sec = five minutes), 6:00:00 (hour:min:sec = six hours)
- F6-F7: Zoom In & Out - Zooms symmetrically around the ruler (if no ruler is present, a ruler will be placed at the center of the Trend Area).
- F8: Area Zoom - With this function you can select a smaller scope of the current scope. After pressing Area Zoom, click once inside the Trend Area to mark the start of the new scope, and click a second time to mark the end of the scope. After clicking the second time, the scope will be recalculated and the Trend Area updated.
- F10: Main - Switches to the Main2 menu.

Scope Adjustment

The Scope Adjustment menu is shown in [Figure 202](#).

| | | | | |
|---------------|---------------|----|---------------|---------------|
| F1 < Ruler | F2 Ruler > | F3 | F4 < Scope | F5 Scope > |
| F6 | F7 | F8 | F9 | F10 Main |

Figure 202. Scope Adjustment Menu

These functions are used for controlling the scope and the ruler:

- F1-F2: Ruler < & > - Ruler Step left and right, has the same function as the Ruler Step Keys next to the Trend Info.
- F4-F5: Scope < & > - Changes scope 20% respectively left and right. It is the same function as the Scope Keys next to the Trend Info.
- F10: Main - Switches to the Main2 menu.

Trace

The Trace menu is shown in [Figure 203](#).

| | | | | |
|-----------------|------------------|--------------------|---------------------|------------------|
| F1 Y max/min | F2 Hide Trace | F3 Time Offset | F4 Configuration | F5 Next Trace |
| F6 | F7 Show Trace | F8 Filter Param | F9 | F10 Main |

Figure 203. Trace Menu

It contains functions for individually setting up each trend:

- F1-F2: Y max/min - Set up the maximum and minimum Y-scale values for the selected trend.
- F2 & F7: Hide/Show Trace - Hide or show each trend. It has the same function as the toggle buttons to the left of the Trend Info.
- F3: Time Offset - Delay or advance a trend in time. A positive value moves the trend to the right in the Trend Area. A negative value moves to the left. The delay time should be entered in the same format as the scope (see [Zoom](#) on page 349 for description of the format). If the Time Offset for a trend is 5:00 (five minutes), placing the ruler at the time 8:00:00 will result in 8:05:00 in the RulerTime field and in the Trend Info. Furthermore, it will result in the value at 8:05:00 rather than at 8:00:00 in the RulerValue field for the trend with a time offset. When a Time Offset value is different from zero, the associated field in the Trend Info is highlighted.
- F4: Configuration - Switches to the Configuration menu.
- F5: Next Trace - Selects the next trend in the Trend Info to be the selected trend.

- F8: Filter Param - Applies the filter function to the selected trend. The filter factor must be a number between 0 and 1. 0 means no filtration and 1 means full filtration.
- F10: Main - Switches to the Main2 menu.

Configuration

The Configuration menu, [Figure 204](#), is mainly used for creating and deleting new trends. The name of the current log (if any) is shown in the edit-field below the menu buttons. To see another log, simply enter the name of the log in the edit-field and press ENTER.

History logs can be referenced using either of the following formats:

B0502
\$HSB0502-1-o

TTD-logs are referenced by the name with which they were mapped in history.

| | | | | |
|----|----|--------------|--------------|--------------|
| F1 | F2 | F3 | F4 | F5 |
| F6 | F7 | F8 Linear | F9 Delete | F10 Trace |

Figure 204. Configuration Menu

The functions provided by this menu are described below:

- F8: Linear/SampleHold - Used to toggle between the two representations of each trend.
- F9: Delete - Deletes a trend from the Trend element and updates the Trend Info.
- F10: Trace - Switches to the Trace menu.

Appendix D Font Scaling Guidelines

Introduction

When you build displays on a screen with greater resolution than the screen where the displays will ultimately run, the text in some display elements (for instance multitext and list display elements) may go out of alignment. If this occurs, you can scale the font sizes to maintain alignment.

The information for scaling font size for PC screen resolution is contained in the *aidfonts.ini* file. This file is placed in the windows subdirectory during the installation procedure for the PC client. The *aidfonts.ini* file contains 18 Basesize values that correspond to the 18 font type/font size combinations available via the Font Select dialog ([Font](#) on page 82). There are six basesize values for each of the three font types (Arial, Times New Roman, Courier). Font types are identified in the *aidfonts.ini* file by a facename parameter. See [Table 65](#).

To make a scaling adjustment to a particular font, you need to know the font number. This is indicated by the Font property in the Properties window for the display element. Having determined the font name, you can look up the corresponding font name and base number in [Table 65](#) below. You use this information in the formula below to determine which basesize value to adjust in the *aidfonts.ini* file. Then you increase the value by one to increase the size slightly, or decrease the value by one to decrease the size slightly.

The formula to determine which basesize to adjust is as follows:

$$\text{Basesize}_n = (\text{Font Number} - \text{Font Base Number}) \text{ Modulo } 6$$

For example, to determine which basesize to adjust when the font type is Courier and the font number is 63:

$$\text{Basesize}_n = (63 - 48) \text{ Modulo } 6 = 3 \quad (\text{Note: Modulo finds the remainder})$$

The facename for courier is Facename_2, so the basesize to adjust is Basesize_3 under Facename_2. If you need to increase the font size, set the value to -20 (up one). If you need to decrease the font size, set the value to -18 (down one).

Table 65. Information for Font Scaling Adjustment

| Facename | Font Name | Font Number Range | Font Base Number | Basesize |
|------------|-----------------|-------------------|------------------|------------------|
| Facename_0 | Arial | 0 to 23 | 0 | Basesize_0 = -11 |
| | | | | Basesize_1 = -14 |
| | | | | Basesize_2 = -17 |
| | | | | Basesize_3 = -18 |
| | | | | Basesize_4 = -23 |
| | | | | Basesize_5 = -25 |
| Facename_1 | Times New Roman | 24 to 47 | 24 | Basesize_0 = -9 |
| | | | | Basesize_1 = -13 |
| | | | | Basesize_2 = -16 |
| | | | | Basesize_3 = -19 |
| | | | | Basesize_4 = -25 |
| | | | | Basesize_5 = -32 |
| Facename_2 | Courier | 48 to 72 | 48 | Basesize_0 = -10 |
| | | | | Basesize_1 = -15 |
| | | | | Basesize_2 = -17 |
| | | | | Basesize_3 = -19 |
| | | | | Basesize_4 = -25 |
| | | | | Basesize_5 = -34 |

Appendix E Batch Display Demonstration

Overview

The batch displays are part of the display demonstration which is described in [Appendix A, A Demonstration of Display Services](#). These displays let you query and display recipe and production data for Batch 300 applications. Production data is collected and stored via the Information Manager Production Data Log (PDL) software package.

You can use these displays as-built, or you can use the displays as the basis for building custom displays that are more suited to your batch display requirements.

The displays reside under a BATCH_EXT.svg in the Display Services directory structure. The file names are:

- DisplayDirectory.svd - See [Display Directory on page 355](#).
- Job_Summary.svd - See [Job Summary on page 357](#).
- Batch_Summary.svd - [Batch Summary on page 360](#).
- Batch_Time_Interval.svd - See [Batch Trend on page 367](#).

Display Directory

The Display Directory, [Figure 205](#), provides page links to these Batch 300 displays:

- [Job Summary](#)
- [Batch Record Variables](#)
- [Batch Trend](#)
- [Lab Data Entry](#)

You can also exit the demo from this display, or go to the main display demonstration.

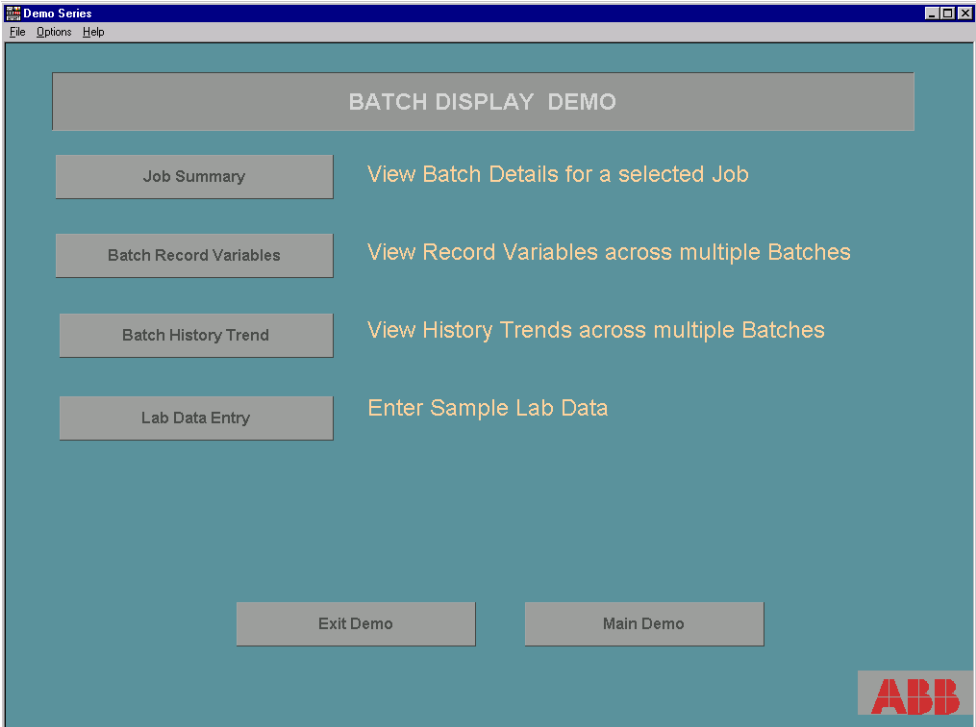


Figure 205. Display Directory

Display Directory Menus

The Display Directory has the following menus. These menus are common to all Batch displays.

| | |
|---------|---|
| File | File - Print: Print the current display File - Exit: Exit Display Services |
| Options | n/a |
| Help | Communication Status About Display Services |

Job Summary

The **Job Summary** button on the [Display Directory](#) opens the Job Summary Report window, [Figure 206](#). This window lets you select a job that you want to generate a job summary report for, and then presents the following information for the job you select:

- Name of selected job
- Recipe for selected job
- Job Start and End Time
- Job Goal
- Job Actual
- Job Tolerance
- List of batches executed for the selected job with corresponding batch start and end times

When you select a batch from the batch list, the corresponding batch goal and batch actual are displayed. You can access further details regarding the selected batch by pressing the **Batch Summary** button in the upper right corner. Details regarding the Batch Summary are provided in [Batch Summary on page 360](#).

This display has the same menu items as the Display Directory. See Display Directory menus in [Display Directory on page 355](#).

When you select a job, data for the selected job is displayed here.

2

First, click here to display Job List. Then select a job from the list.

1

This Time Filter lets you narrow the Job List when the list is too long

6

The screenshot shows the 'Job Summary Report' window. At the top, there's a menu bar with 'File', 'Options', and 'Help'. Below it, a title bar says 'BATCH EXT Job Summary'. The main area is divided into sections. On the right, there's a 'Return to Main' button and a 'Batch Summary' button. Below these are input fields for 'Job Name' (J97205_001), 'Job Creation Time' (19-SEP-1997 09:31:01), 'Start Time' (30-SEP-1997 13:06:02), and 'End Time' (30-OCT-1997 13:06:02). A date format 'DD-MON-YYYY HH24:MI:SS' is shown. Below these are fields for 'Selected Job' (J97205_001), 'Selected Batch' (342246DF0300_BATCH_IN_JOB_J97205_001), 'Start Time' (19-SEP-1997 09:31:01), 'End Time' (19-SEP-1997 09:37:14), 'Recipe' (AID_UNITS), 'Job Goal' (1.00), 'Job Actual' (17.00), and 'Tolerance' (0.00). At the bottom, there's a table of batches with columns 'Batch Identifier', 'Batch Name', 'Start Time', and 'End Time'. The fifth row is highlighted in blue. On the far right, there are buttons for 'Batch Goal' and 'Batch Actual'.

Job Summary Report

Return to Main

Batch Summary

Job Name J97205_001 Job Creation Time 19-SEP-1997 09:31:01

DD-MON-YYYY HH24:MI:SS

Start Time 30-SEP-1997 13:06:02

End Time 30-OCT-1997 13:06:02

Selected Job J97205_001 Selected Batch 342246DF0300_BATCH_IN_JOB_J97205_001

Start Time 19-SEP-1997 09:31:01 Job Goal 1.00

End Time 19-SEP-1997 09:37:14 Job Actual 17.00

Recipe AID_UNITS Tolerance 0.00

| Batch Identifier | Batch Name | Start Time | End Time |
|------------------|-------------------------|----------------------|----------------------|
| 3422463F0300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:31:01 | 19-SEP-1997 09:32:20 |
| 342246720300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:31:54 | 19-SEP-1997 09:33:15 |
| 342246A90300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:32:49 | 19-SEP-1997 09:34:09 |
| 342246DF0300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:33:44 | 19-SEP-1997 09:35:04 |
| 342247160300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:34:39 | 19-SEP-1997 09:35:59 |
| 3422474D0300 | BATCH_IN_JOB_J97205_001 | 19-SEP-1997 09:35:33 | 19-SEP-1997 09:36:51 |

Batch Identifier Batch Name Start Time End Time

3422463F0300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:31:01 19-SEP-1997 09:32:20

342246720300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:31:54 19-SEP-1997 09:33:15

342246A90300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:32:49 19-SEP-1997 09:34:09

342246DF0300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:33:44 19-SEP-1997 09:35:04

342247160300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:34:39 19-SEP-1997 09:35:59

3422474D0300 BATCH_IN_JOB_J97205_001 19-SEP-1997 09:35:33 19-SEP-1997 09:36:51

Batch Goal

Batch Actual

3

4

5

A list of batches for selected Job is displayed here.

Select a batch to display Batch Goal and Batch Actual values here.

Click here to get further details regarding the selected batch

Figure 206. Job Summary Report

The display controls are described below:

| | |
|----------------|--|
| Job List | The Job List is a drop down list which is displayed by clicking the associated button at the far right end of the list. Job Name and Job Creation Time are column headings for list of jobs. |
| Start Time | Start time to filter jobs in Job List when the time filter is active. |
| End Time | End time to filter jobs in Job List when the time filter is active. |
| TIME FILTER | <p>The TIME FILTER button is used to activate or deactivate the time filter for the Job List. The time filter is active when the TIME FILTER label is green. The time filter is deactivated when the label is black. Click on the drop down button to display a time filtering menu with the following choices:</p> <p>CURRENT: makes Start Time & End Time current date & time 24 HRS: End Time is current, Start Time is 24 hours back 7 DAYS: End Time is current, Start Time is 7 days back 30 DAYS: End Time is current, Start Time is 30 days back</p> |
| Selected Job | Name of selected job (BLANK if no job is selected). |
| Start Time | Date and time when selected job was started. |
| End Time | Date and time when selected job ended. |
| Recipe | Recipe for selected job |
| Selected Batch | Name of batch selected from Batch list. |
| Job Goal | Job Goal for selected job. |
| Job Actual | Job Actual for selected job. |
| Tolerance | Job Tolerance for selected job. |
| Batch Goal | Batch Goal for selected batch. |
| Batch Actual | Batch Actual for selected batch. |
| Lower Window | Batch Identifier, Batch Start Time, and Batch End Time |
| Return to Main | Click to return to the Batch Display Directory. |
| Batch Summary | Click to display the Batch Summary report for the selected batch. |

Batch Summary

To display a Batch Summary report, using the [Job Summary](#) Report window, first select a batch within the selected job, and then click the **Batch Summary** button.

A typical Batch Summary report is shown in [Figure 207](#). The following data is displayed for the selected batch:

- The name of the job that the batch was executed in.
- Batch start and end time.
- Batch Goal, Yield and tolerance
- Batch recipe
- A list of units where phases were run for the batch with start and endtimes
- When a unit is selected from the list, the Phase window presents a list of phases that ran on that unit with their start and end times.
- When a phase is selected, the Phase parameter window presents a list of parameters for the selected phase, and all Recorded data within a phase.

This display also provides page links for the Job Summary, Display Directory, and Batch Record Variables functions.

This display has the same menu items as the Directory Display. See Display Directory menus in [Display Directory on page 355](#).

When you open the Batch Summary Report window, data for the batch you selected via the Job Summary Report window is displayed here.

1 Batch Summary Report

Job Name J97205_001 Selected Batch Name 342247160300_BATCH_IN_JOB_J97205_001
 Batch Start Time 19-SEP-1997 09:34:39 Batch Goal 4.00 DD-MON-YYYY HH24:MI:SS
 Batch End Time 19-SEP-1997 09:35:59 Batch Yield 4.00 Start Time 27-JAN-1998 13:42:25
 Recipe AID_2UNITS Tolerance 0.00 End Time 27-JAN-1998 13:42:25

2 Units

| Unit | Start Time | End Time |
|------------|----------------------|----------------------|
| TEST1_UNIT | 19-SEP-1997 09:34:39 | 19-SEP-1997 09:35:06 |
| TEST2_UNIT | 19-SEP-1997 09:35:06 | 19-SEP-1997 09:35:32 |
| TEST3_UNIT | 19-SEP-1997 09:35:06 | 19-SEP-1997 09:35:32 |
| TEST4_UNIT | 19-SEP-1997 09:35:06 | 19-SEP-1997 09:35:58 |

Phase name Phase start time Phase end time

| Phase name | Phase start time | Phase end time |
|------------|----------------------|----------------------|
| AID-TEST | 19-SEP-1997 09:35:35 | 19-SEP-1997 09:35:36 |
| AID-TEST | 19-SEP-1997 09:35:36 | 19-SEP-1997 09:35:36 |

Phase name Variablename Value Recorded time

| Phase name | Variablename | Value | Recorded time |
|------------|--------------|---------------------------|----------------------|
| AID-TEST | RPTSTR1 | 09:35_09/19 TEST4_UNIT | 19-SEP-1997 09:35:35 |
| AID-TEST | VALUE001 | PHASE-UNIT 4 SECOND CHAIN | 19-SEP-1997 09:35:36 |
| AID-TEST | RPTSTR1 | 09:35_09/19 TEST4_UNIT | 19-SEP-1997 09:35:36 |

CCF Messages For PHASE: AID-TEST MESSAGE TYPE CCF

3 When you select a unit, a list of phases for the selected unit is displayed here.

4 When you select a phase, parameters & recorded variables for the selected phase are shown here.

5 When a message occurs for the selected phase, the message is displayed here. You can select the type of message (CCF, TCL, or ALL) using the drop down menu.

Figure 207. Batch Summary Report

The display controls are described below:

| | |
|---------------------|---|
| Job Name | Name of Job that was selected on Job Summary report. |
| Selected Batch Name | Name of batch that was selected on Job Summary report. |
| Batch Start Time | Start time for selected batch. |
| Batch End Time | End time for selected batch. |
| Recipe | Recipe for selected job. |
| Batch Goal | Batch goal for selected batch. |
| Batch Yield | Batch yield for selected batch. |
| Tolerance | Batch tolerance for selected batch. |
| Start Time | Start time to filter Batch List when the time filter is active. |
| End Time | End time to filter Batch List when the time filter is active. |
| TIME FILTER | <p>The TIME FILTER button is used to activate or deactivate the time filter for the Batch List. The time filter is active when the TIME FILTER label is green. The time filter is deactivated when the label is black. Click on the drop down button to display a time filtering menu with the following choices:</p> <p>CURRENT: makes Start Time & End Time current date & time</p> <p>24 HRS: End Time is current, Start Time is 24 hours back</p> <p>7 DAYS: End Time is current, Start Time is 7 days back</p> <p>30 DAYS: End Time is current, Start Time is 30 days back</p> |
| Unit Window | <p>This window provides a list of units where phases ran for the selected batch. The information provided for each unit is:</p> <p>Unit Name</p> <p>Start Time</p> <p>End Time</p> |
| Phase Window | <p>When you select a unit from the unit list, this window provides a list of phases that ran on the selected unit. The information provided for each phase is:</p> <p>Phase Name</p> <p>Phase Start Time</p> <p>Phase End Time</p> |

| | |
|------------------------|--|
| Phase Variable Window | When you select a phase from the phase list, this window shows the recorded variables (from PDL_VARIABLE_VIEW) for the selected phase. The information provided is: Phase Name Variable Name Value Recorded Time |
| Message Type | Button displays a pull down menu with the following choices: ALL: Show all messages. CCF: Show just CCF messages. TCL: Show just TCL messages. |
| Message Window | Messages of the selected type are displayed in this window as they occur. |
| Return to Main | Click to return to the Batch Display Directory. |
| Job Summary | Click to display the Job Summary report. |
| Multiple Batch Details | Click to display the Multiple Batch Data window, Batch Record Variables on page 363. |

Batch Record Variables

The **Batch Record Variables** button on the [Display Directory](#) opens the Multiple Batch Data window where you can execute queries to get recorded data for multiple batches, [Figure 208](#).



You can also open this window by clicking the **Multiple Batch Details** button on the Batch Summary Report.

The data source is the PDL_VARIABLE_VIEW. Refer to *System 800xA Information Management Data Access and Reports (3BUF001094*)* for details regarding the contents of this view. Using the buttons on this window you can set up a query to return:

- All recorded data in PDL_VARIABLE_VIEW.



Depending on the size of the database, this can take a significant amount of time. Using a 50/60 MHz IMS takes about 12 seconds to access 3914 pieces of data.

To set up the filter to view all data, click on the **ALL** button for JOB, and then click on the **ALL** button for VARIABLE. Leave all other filter parameters at their default settings.

- Information filtering by JOB.
- Information filtering by JOB-BATCH.
- Information filtering by JOB-BATCH-UNIT.
- Information filtering by JOB-BATCH-UNIT-PHASE.
- Information filtering by JOB-BATCH-UNIT-PHASE-Variable name.

This display also lets you access recorded data across multiple batches within a job, or recorded data across multiple jobs. You can then filter to access a specific variable name across multiple jobs/batches.

All of the above filters can be filtered by time using the TIME FILTER.

From a PC, you can copy and paste data into another application such as Excel for further analysis and reporting.

1 Select filtering parameters using these drop down menus.

2 Click on QUERY after you specify the filter.

The screenshot shows the 'Multiple Batch Data' window with the following controls and data:

Filtering Controls:

- JOB:** J97205_001
- BATCH:** (empty)
- UNIT:** (empty)
- PHASE:** (empty)
- VARIABLE:** VALUE001
- Buttons:** ALL (for each field)
- Start Time:** 31-OCT-1997 09:18:25
- End Time:** 31-OCT-1997 09:18:25
- TIME FILTER:** CURRENT
- Variables across BATCHES:** VALUE001
- QUERY:** (button)

Table Data:

| BATCH | UNIT | PHASE | VARIABLENAME | VALUE |
|--------------------------------------|------------|----------|--------------|--------------------------|
| 3422463F0300_BATCH_IN_JOB_J97205_001 | TEST1_UNIT | AID-TEST | VALUE001 | PHASE-UNIT1 |
| 3422463F0300_BATCH_IN_JOB_J97205_001 | TEST2_UNIT | AID-TEST | VALUE001 | PHASE-UNIT2 |
| 3422463F0300_BATCH_IN_JOB_J97205_001 | TEST3_UNIT | AID-TEST | VALUE001 | PHASE-UNIT3 |
| 3422463F0300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 FIRST CHAIN |
| 3422463F0300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 SECOND CHAIN |
| 342246720300_BATCH_IN_JOB_J97205_001 | TEST1_UNIT | AID-TEST | VALUE001 | PHASE-UNIT1 |
| 342246720300_BATCH_IN_JOB_J97205_001 | TEST2_UNIT | AID-TEST | VALUE001 | PHASE-UNIT2 |
| 342246720300_BATCH_IN_JOB_J97205_001 | TEST3_UNIT | AID-TEST | VALUE001 | PHASE-UNIT3 |
| 342246720300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 FIRST CHAIN |
| 342246720300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 SECOND CHAIN |
| 342246A90300_BATCH_IN_JOB_J97205_001 | TEST1_UNIT | AID-TEST | VALUE001 | PHASE-UNIT1 |
| 342246A90300_BATCH_IN_JOB_J97205_001 | TEST2_UNIT | AID-TEST | VALUE001 | PHASE-UNIT2 |
| 342246A90300_BATCH_IN_JOB_J97205_001 | TEST3_UNIT | AID-TEST | VALUE001 | PHASE-UNIT3 |
| 342246A90300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 FIRST CHAIN |
| 342246A90300_BATCH_IN_JOB_J97205_001 | TEST4_UNIT | AID-TEST | VALUE001 | PHASE-UNIT4 SECOND CHAIN |
| 342246DF0300_BATCH_IN_JOB_J97205_001 | TEST1_UNIT | AID-TEST | VALUE001 | PHASE-UNIT1 |
| 342246DF0300_BATCH_IN_JOB_J97205_001 | TEST2_UNIT | AID-TEST | VALUE001 | PHASE-UNIT2 |

NO. OF RECORDS 30

3 Column headings based on variable list displayed here.

4 Variable data displayed here.

Figure 208. Multiple Batch Data Window

The display controls are described below:

| | |
|------------------------------------|--|
| JOB | Button displays a list of available jobs. You must specify a job for the query, or select the ALL button. |
| BATCH | Button displays a list of batches for the selected job. You may specify a batch for the query (optional). |
| UNIT | Button displays a list of units for the selected batch. You may specify a unit for the query (optional). |
| PHASE | Button displays a list of phases for the selected unit. You may specify a phase for the query (optional). |
| VARIABLE | Button displays a list of variables for selection criteria (as defined above). You must specify a variable for the query. |
| Start Time | Start time to apply to other filters when the time filter is active. |
| End Time | End time to apply to other filters when the time filter is active. |
| TIME FILTER | <p>The TIME FILTER button is used to activate or deactivate the time filter. The time filter is active when the TIME FILTER label is green. The time filter is deactivated when the label is black. Click on the drop down button to display a time filtering menu with the following choices:</p> <p>CURRENT: makes Start Time & End Time current date & time</p> <p>24 HRS: End Time is current, Start Time is 24 hours back</p> <p>7 DAYS: End Time is current, Start Time is 7 days back</p> <p>30 DAYS: End Time is current, Start Time is 30 days back</p> |
| Variables Across JOBS (or BATCHES) | This filter is used to access recorded data across multiple batches within a job, or recorded data across multiple jobs. You can then filter to access a specific variable name across multiple jobs/batches. This filter cancels out any batch, unit, phase, and variable specifications in the other filters. |
| Query | Click this button to execute the query based on the filter you specified using the buttons described above. |

| | |
|------------------------|--|
| Column Headings Window | The column headings window remains blank until you execute the query. Column headings are determined by the variables returned by the query. |
| Variables Window | Variable data returned by the query is displayed here. |
| NO. of Records | This field indicates the number of records from the PDL_VARIABLE_VIEW returned by the query. |
| Return to Main | Click to return to the Batch Display Directory. |
| Job Summary | Click to display the Job Summary report. |
| Batch Trend | Click to display the Batch History Trend Variables display, Batch Trend on page 367 . |

Batch Trend

The **Batch History Trend** button on the [Display Directory](#) starts a dialog for specifying batch data to display on a trend display. Clicking the **Batch History Trend** button opens the Batch Data Select window, [Figure 209](#). This is where you specify the batch data to plot on the trend display. This window has the following fields and buttons to let you select one variable in as many as four batches to display on a trend graph:

| | |
|--------------|--|
| | Job Name Filter - Enter a filter for a job name. % means all jobs. |
| Job Filter | Job Start Time Filter Job End Time Filter RESET - Resets the Job Filter fields to their respective defaults. |
| List of Jobs | This button displays a drop down list of jobs based on the job filtering parameters specified above. |
| Selected Job | Name of the job you selected from the List of Jobs. |

| | |
|--------------------------------|---|
| | Batch Name Filter |
| | Batch Start Time Filter |
| Batch Filter | Batch End Time Filter |
| | RESET - Resets the Batch Filter fields to their respective defaults. |
| List of Batches | This window provides a list of batches based on the selected job, and the batch filtering parameters specified above. |
| Selected Batches | Four fields available to display names of up to four batches that you select from the List of Batches |
| List of Trend ON/OFF Variables | This window provides a list of Trend ON/OFF variables for the batch(es) you select from the List of Batches. |
| Selected Variable | Name of the variable you selected from the List of TREND ON/OFF Variables. |
| Batch Demo Menu | Click to return to the Batch Display Directory. |
| Batch Data Plot | Click to generate a plot of the selected batch data, Lab Data Entry on page 370. |
| Date | Current date. |
| Time | Current time. |

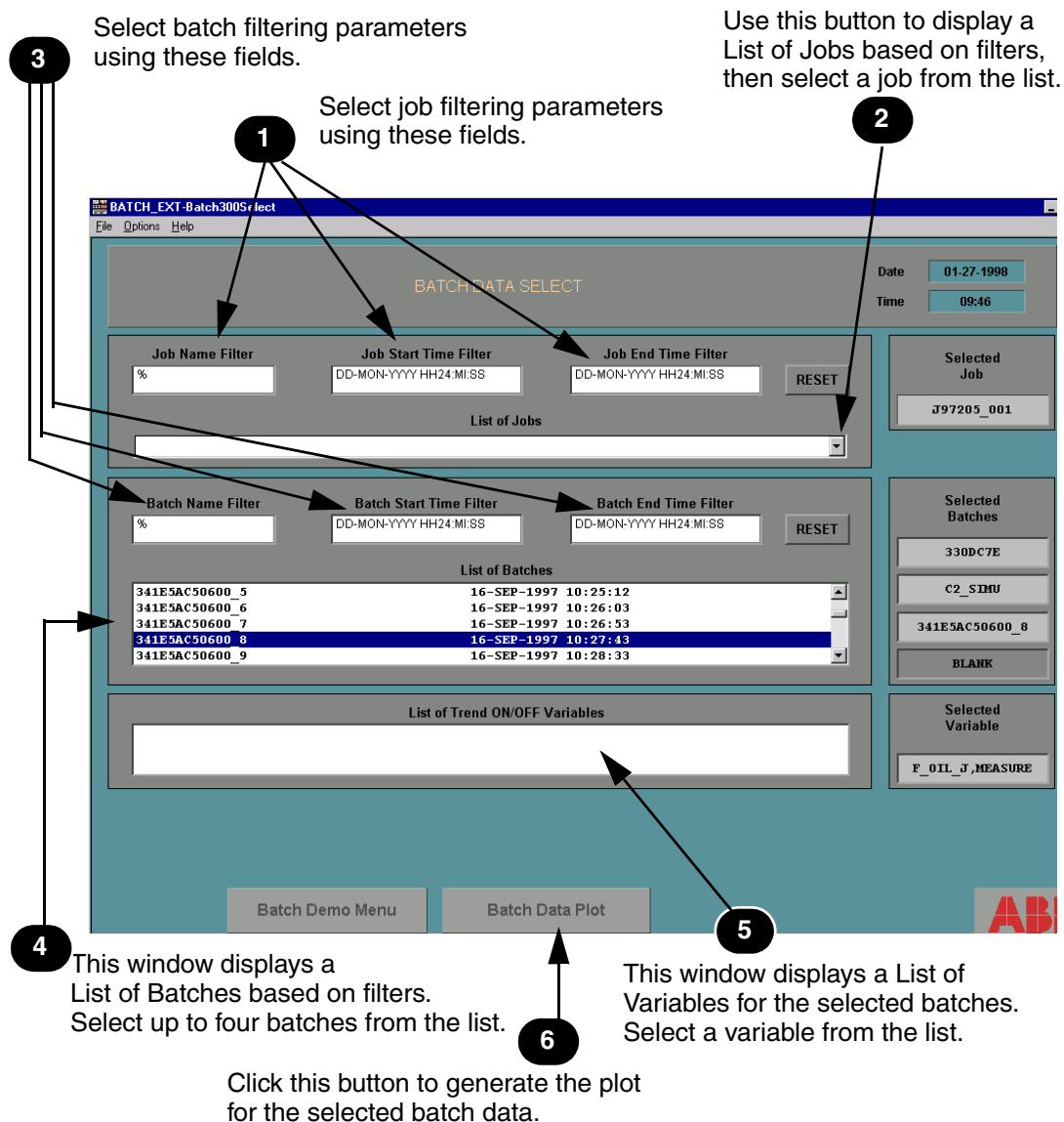


Figure 209. Batch Data Select Window

Lab Data Entry

This window shows a plot of the batch data you selected via the Batch Data Select window, [Batch Trend on page 367](#). A typical Batch Data Plot is shown in [Figure 210](#).

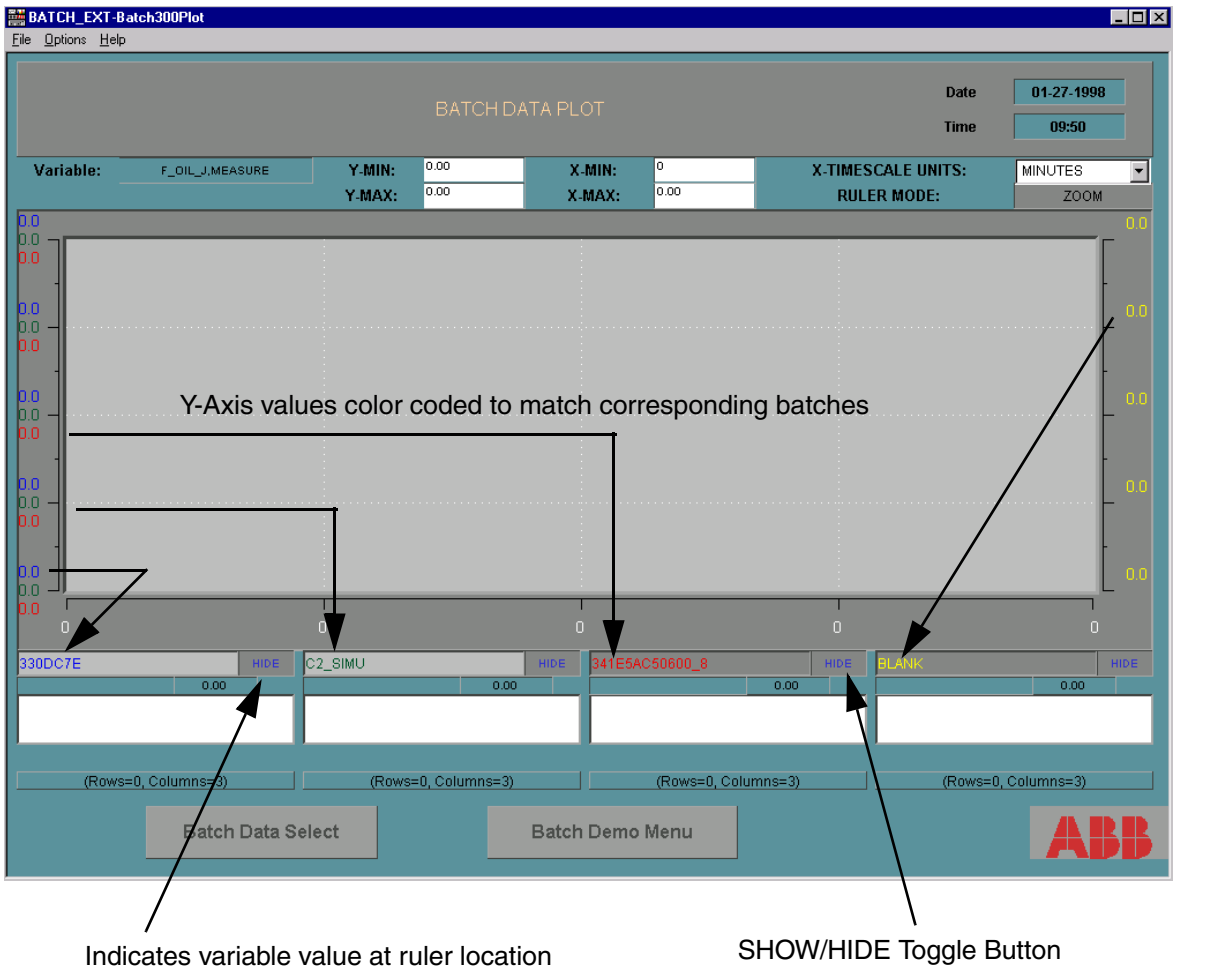


Figure 210. Batch Data Plot

The Y-axis labels are color-coded to match the color of the corresponding batch.

This window has the following fields and buttons for setting up and displaying the selected batch data:

| | |
|-------------------------|---|
| Variable | Name of variable selected via Batch Data Select window. |
| Y-MIN | Bottom limit of Y axis. |
| Y-MAX | Top limit of Y axis. |
| X-MIN | Bottom limit of X axis. |
| X-MAX | Top limit of X axis. |
| X-TIMESCALE UNITS | Button displays a drop down menu with the following choices: SECONDS MINUTES HOURS DAYS |
| RULER MODE | ZOOM |
| Batch Information Block | Each of the batches selected via the Batch Data Select window (up to four batches) has a dedicated information block that provides the following information: <ul style="list-style-type: none">- Batch name- variable value based on ruler location- SHOW/HIDE button to either show or hide the trend plot for that batch |
| Batch Demo Menu | Click to return to the Batch Display Directory. |
| Batch Data Select | Click to return to the Batch Data Select window, Batch Trend on page 367 . |
| Date | Current date. |
| Time | Current time. |

Revision History

Introduction

This section provides information on the revision history of this User Manual.



The revision index of this User Manual is not related to the 800xA 6.0 System Revision.

Revision History

The following table lists the revision history of this User Manual.

| Revision Index | Description | Date |
|----------------|---------------------------------------|----------------|
| - | First version published for 800xA 6.0 | August 2014 |
| A | Updated for 800xA 6.0.3 | September 2016 |

Updates in Revision Index A

The following table shows the updates made in this User Manual for System 800xA 6.0.3:

| Updated Section/Sub-section | Description of Update |
|--|---|
| Introduction and Display Services Configuration sections | Pages 20 and 224 updated as per the markups received. |

Index

A

ABBBButton 155
acos 290
action 286
Action functions 288
Action script 224
Activate 148
activeX 216
add/modify log entry 139
Addition 282
aid color palette 81
aid fonts 83
Align 92
align options 93
aligning elements 66
ALM 184
applications 70
ASCII 290 to 291
asin 291
asql 291
atan 292
available licenses 29

B

Background 221
Bar 157
bit 292
Bitmap 111
bitmap images 23
BitmapPath 110
broadcast 293
broadcast action 90
build 29
Build mode 19
Button 152

button type 155
button visible 155

C

ChangeScope 189
Check Syntax 79
choices list 79
chr 293
ClearLogData 190
client 17
color 80
color palette 81
colors
 aid 81
 Windows 80
columns 294
Combobox 149
comment 283
communication timeout 23
configuring properties 40
constants 278
control flow 283
Copy
 Property Definitions 77 to 78
Copy, Edit Menu 64
cos 294
cosine 294
Cut, Edit Menu 63

D

data function 132
data statement 44
DataChanged action 226
DataQuery 114, 125, 148, 222
DataSource, trend 180

- dcsgetobjinfo 300
- dcssub 300, 306
- dec 302
- defining a query 43
- delete 101, 302
- Delete, Edit Menu 64
- Description property 88
- dialog 303
- display 303
 - adding a new display 36, 74
 - group 74
 - name 37
 - new 36
 - properties 87
 - server status 28
- display caching 22
- display client as ActiveX control 30
- display elements
 - ABBBButton 155
 - active x 216
 - align 92
 - bar 157
 - button 152
 - combobox 149
 - delete 101
 - distribute 94
 - edit 145
 - gauge 197
 - group 94
 - grouping 94
 - inserting a new display element 38
 - list 126
 - matrix 131
 - multibar 162
 - multitext 122
 - name 219
 - numeric 115
 - ordering 95
 - pie 167
 - polygon 106

- polyline 105
- rename 101
- shape 108
- smartShade 201
- text 112
- timer 103
- trend 170
- xyPlot 209
- displayexist 304
- DisplayIE 30
- distribute 94
- distribute elements 66
- Division 282
- dlovar 304
- Do-While 284
- dsevar 305

E

- Edit 145
- Edit menu 63
- Editable 152
- element list 97
- Environment variables 21
- Equipment Requirements 20
- escape character 281
- event property 78
- execute 305
- execute command 226
- execution order 95, 100
- exit 46, 306
- External Editor 79

F

- File menu 62
- Filter 195
- float constant 281
- font 82
- fonts
 - aid 83
 - windows 82

Foreground 221
Foreground, ABBButton 157
Format 118
Framestate 221
Framewidth 221
FuncKeyElement 89

G

gauge 197
GaugeLocation & -Color 200
General Purpose functions 290
Geometry properties 218
getenv 307
GetLoggedAttributes 190
GotFocus 148
Grid 91
Grid settings 91
group 36
group elements 67
grouping 94

H

Height 129, 219
hex 308
history log entry 139
host 25

I

if-then-else-endif 283
iif 308
indexed referencing 126
Indexing 126
insert 308
instr 309
integer constant 280
isevar 309
ishost 310
ItemDoubleClicked 130
ItemSelected 152

L

Last Check Window 79
last history value 115, 120 to 121
length 310
List 126
Location, pie 170
log files 23
log in 24
LOGMGR 181
lower 310
lpad 311
lsyvar 311
ltrim 311

M

main menu 62
Major- & MinorTickNumber 200
Math functions 289
matrix 131
matrix status 143
MDI 25, 29
member property 77
Menu 89
Menu Bar 62
menus
 change 65
 custom 84, 89
 edit 63
 file 62
 user 65
 view 64
 window 67
method property 77
Modulus 282
Motif frame 221
mouse buttons 58
multibar 162
Multiplication 282
Multistring 280
MultiText 122

multitext 122
multitext dataQuery 125

N

name 37
Name, element 219
Needle, gauge 201
Normal mode, trend 187
notation, numeric 119
Numeric 115
numeric field width 119
numeric format 118
numeric format flags 118
numeric precision 119
Numeric, bar value 161

O

Object Browser 58
object browser 58
OnEntry 88
OnLeave 88
OnPreQuery 88
OPC object 140
Order 95
OS system variable files 286
oscolor 312
osfont 312
ostreatment 313

P

paste buffer 64
Paste, Edit Menu 64
pie 167
polygon 107
polyline 105
print 313
properties 40
 ABBBButton 156
 activeX 217
 bar 158

button 154
category 77
combobox 150
configuration 77
dialog 76
edit 146
event 78
gauge 198
list 128
matrix 132
member 77
method 77
multibar 163
multitext 124
numeric 116
pie 169
polygon 107
polyline 106
property definition window 78
shape 109
smartShade 203
text 113
trend 173
xyPlot 211
properties dialog 75
properties, display element 74
Property Definition Window 42, 78
push, button 155

Q

query 285
Query functions 287
QueryPlusYSelected 191
QuerySelected 191
QueryValues 190
QueryValuesPlusYMinMax 191

R

rand 314
random 314

rearrange 314
rename 101
replace 315
Representation 192
ReQuery 131, 285, 306
requery 225
ResetYValuesSelected 191
return 315
rpad 315
rtrim 316
Ruler 185
run mode 19, 45

S

Scale, gauge 201
Scale, vertical bar 161
Scope, trend 179
SDI 25, 29
SelectedValue 130
server 17
server status 27
Session variables 279, 287
setfocus 316
Shape 108
shape 108
sin 317
smartShade 201
Sort order 166
SQL 183
sql 317
SQL queries
 numeric log data 70
sqrt 318
start-up
 display client 24
statement 278
status 143, 149
Status, toggle button 155
string constant 281
String functions 288

subscribe 300
substr 318
substring 318
Subtraction 282
syntax checker 44, 79
system 319
System variable functions 286

T

tan 319
tangent 292, 319
TCL Unit Array 135
text 112
time 320
time-format 280
TimeOffset 196
timer 103
timer action 105
timer delay 105
timer mode 104
timer properties 104
toggle, button 155
toolbar 61
trace 323
trend 170
 filter 195
Trend, bar graph 161
TrendMode 186
 ChangeScope 189
 ClearLogData 190
 GetLoggedAttributes 190
 normal 187
 QueryPlusYSelected 191
 QuerySelected 191
 QueryValues 190
 QueryValuesPlusYMinMax 191
 ResetYValuesSelected 191
 Update Traces 187
 ZoomArea 188
 ZoomInOut 187

ZoomTimeDuration 188 to 189
Tutorial 35, 335, 339
Type conversions 283
Type, button 155

U

update 323
update statement 225
UpdateTraces 187
upper 324
user element 227
 type 228
user element cache 23
user interface 57
user preference
 DATARETRIEVAL 121
user property 89

V

Value 115, 126
Variables 278
Visible, element 220
VisibleItemCount 152

W

Width 219
windows color palette 80
windows fonts 82
write
 history log 139
 OPC object 140
 process object 139

X

Xpos 218
xyPlot 209

Y

YPos 219

Z

ZoomArea 188
ZoomInOut 187
ZoomTimeDuration 188
ZoomTimeTime 189
z-order 95, 97, 100

Contact us

www.abb.com/800xA
www.abb.com/controlsystems

Copyright© 2016 ABB.
All rights reserved.

3BUF001093-600 A

Power and productivity
for a better world™

