

Applixware Builder Methods Reference

COPYRIGHT NOTICE ON THE VERSION 4.43 SOFTWARE
©1990 - 2010 Vistasource, Inc. All Rights Reserved.

Vistasource, Inc. prepared the information contained in this document for use by Vistasource personnel, customers, and prospects. Vistasource reserves the right to change the information in this document without prior notice. The contents herein should not be construed as a representation or warranty by Vistasource. Vistasource assumes no responsibility for any errors that may appear in this document.

The Proximity Thesauri ®
©1985 Merriam-Webster Inc.
©1988 Williams Collins Sons & Co. Ltd.
©1989 Van Dale Lexicografie bv. ©1989 Nathan. ©1989 Kruger.
©1989 Zanichelli. ©1989 International Data Education a s.
©1989 C.A. Stromber A B. ©1989 Espasa-Calpe.
©1983-1996. Proximity Technology, Inc.
All Rights Reserved.

The Proximity Linguibase And Hyphenation Systems®
©1983 Merriam-Webster Inc.
©1984, 1985, 1986, 1988, 1990 Williams Collins Sons & Co. Ltd.
©1987, 1989 Van Dale Lexicografie bv. ©1988 Munksgaard International Publishers Ltd.
©1988, 1989 International Data Education a s.
©1983-1996 Proximity Technology, Inc.
All Rights Reserved

The Applixware Graphics Filter Pack contains elements of the Generator Metafile Development Libraries (MDL/G)
©1988-1996 Henderson Software, Inc.
All Rights Reserved

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraphs (c) (1) (ii) of SFARS 252.277-7013, or in FAR 52.227-19, as applicable.

Hardware and software products mentioned herein are used for identification purposes only and may be trademarks of their respective companies.

Applixware is a registered trademark of Vistasource, Inc. Applixware, Applixware Real Time, Applixware Data, and Applixware Builder are trademarks of Vistasource, Inc.

This manual was produced using Applixware.

Printed: June 2010

<- attach_to_task@

Attaches object to document task

Class GraphicsClass set

Format flag = this.attach_to_task@(id)

Arguments id The task ID of the document.

Description The set method `attach_to_task@` attaches the GraphicsClass object to the open Graphics document, given the task ID of the document. The method returns a Boolean value, TRUE if the object successfully attaches to the document task, FALSE otherwise. For example, an object will not attach to task if a non-Graphics application task ID is passed to the method.

See [GraphicsClass Methods](#) for more information.

<- attach_to_task@

Attaches object to document task

Class WordsClass set

Format flag = this.attach_to_task@(id)

Arguments id The task ID of the document.

Description The set method `attach_to_task@` attaches the WordsClass object to the open Words document, given the task ID of the document. The method returns a Boolean value, TRUE if the object successfully attaches to the document task, FALSE otherwise. For example, an object will not attach to task if a non-Words application task ID is passed to the method.

See [WordsClass Methods](#) for more information.

<- attach_to_task@

Attaches object to document task

Class SpreadsheetsClass set

Format flag = this.attach_to_task@(id)

Arguments id The task ID of the document.

Description The set method attach_to_task@ attaches the SpreadsheetsClass object to the open Spreadsheets document, given the task ID of the document. The method returns a Boolean value, TRUE if the object successfully attaches to the document task, FALSE otherwise. For example, an object will not attach to task if a non-Spreadsheets application task ID is passed to the method.

See [SpreadsheetsClass Methods](#) for more information.

-> is_windowless@

Sets windowless state of object

Class GraphicsClass set

Format this.is_windowless@(flag)

Arguments flag A Boolean value, FALSE displays the Applixware Graphics window, TRUE suppresses display of the Applixware Graphics window. The Applixware Graphics window for the object is not displayed by default.

Description The set method is_windowless@ sets the windowless state of the application object. If the method is set to FALSE an Applixware Graphics window is displayed upon the first method call to the object. Call the is_windowless@ method before using the object, such as in the initialize_event. The method will have no effect after the first call to the object.

See [GraphicsClass Methods](#) for more information.

-> is_windowless@

Sets windowless state of object

Class WordsClass set

Format this.is_windowless@(flag)

Arguments flag A Boolean value, FALSE displays the Applixware Words window, TRUE suppresses display of the Applixware Words window. The Applixware Words window for the object is not displayed by default.

Description The set method `is_windowless@` sets the windowless state of the application object. If the method is set to FALSE an Applixware Words window is displayed upon the first method call to the object. Call the `is_windowless@` method before using the object, such as in the `initialize_event`. The method will have no effect after the first call to the object.

See [WordsClass Methods](#) for more information.

-> `is_windowless@`

Sets windowless state of object

Class SpreadsheetsClass set

Format `this.is_windowless@(flag)`

Arguments flag A Boolean value, FALSE displays the Applixware Graphics window, TRUE suppresses display of the Applixware Graphics window. The Applixware Graphics window for the object is not displayed by default.

Description The set method `is_windowless@` sets the windowless state of the application object. If the method is set to FALSE an Applixware Graphics window is displayed upon the first method call to the object. Call the `is_windowless@` method before using the object, such as in the `initialize_event`. The method will have no effect after the first call to the object.

See [SpreadsheetsClass Methods](#) for more information.

-> `menu_bar_id@`

Sets document menu bar

Class GraphicsClass set

Format `this.menu_bar_id@(id, mbInfo)`

Arguments id A unique number identifying the custom menu bar for the application window. id can be in the range of 21 to 499:

21-99	Reserved for dialog boxes.
100-199	Reserved for Spreadsheet menu bar.
200-299	Reserved for Words menu bar.
300-399	Reserved for Graphics menu bar.

400-499 Reserved for Macro editor menu bar.

id cannot be a number between 0 and 20, as those numbers are hard-coded for standard Applixware menu bars listed in the app_ids_.am file in your axlocal/elf directory.

mbInfo The name of the file containing the menu bar definition, or an array with the actual menu bar definition.

Description The set method menu_bar_id@ sets the menu bar for the application window. The method changes the menu bar for an open spreadsheet, or initializes the menu bar for a spreadsheet that is closed.

See [GraphicsClass Methods](#) for more information.

-> menu_bar_id@

Sets document menu bar

Class WordsClass set

Format this.menu_bar_id@(id, mbInfo)

Arguments id A unique number identifying the custom menu bar for the application window. id can be in the range of 21 to 499:

21-99	Reserved for dialog boxes.
100-199	Reserved for Spreadsheet menu bar.
200-299	Reserved for Words menu bar.
300-399	Reserved for Graphics menu bar.
400-499	Reserved for Macro editor menu bar.

id cannot be a number between 0 and 20, as those numbers are hard-coded for standard Applixware menu bars listed in the app_ids_.am file in your axlocal/elf directory.

mbInfo The name of the file containing the menu bar definition, or an array with the actual menu bar definition.

Description The set method menu_bar_id@ sets the menu bar for the application window. The method changes the menu bar for an open spreadsheet, or initializes the menu bar for a spreadsheet that is closed.

See [WordsClass Methods](#) for more information.

-> menu_bar_id@

Sets document menu bar

Class SpreadsheetsClass set

Format this.menu_bar_id@(id, mblInfo)

Arguments

id	A unique number identifying the custom menu bar for the application window. id can be in the range of 21 to 499:
21-99	Reserved for dialog boxes.
100-199	Reserved for Spreadsheet menu bar.
200-299	Reserved for Words menu bar.
300-399	Reserved for Graphics menu bar.
400-499	Reserved for Macro editor menu bar.

id cannot be a number between 0 and 20, as those numbers are hard-coded for standard Applixware menu bars listed in the app_ids_.am file in your axlocal/elf directory.

mblInfo The name of the file containing the menu bar definition, or an array with the actual menu bar definition.

Description The set method menu_bar_id@ sets the menu bar for the application window. The method changes the menu bar for an open spreadsheet, or initializes the menu bar for a spreadsheet that is closed.

See [SpreadsheetsClass Methods](#) for more information.

<- error_event

Called for system errors

Class GraphicsClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error

handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [GraphicsClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class GraphicsClass event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [GraphicsClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class GraphicsClass event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [GraphicsClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class GraphicsClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [GraphicsClass Methods](#) for more information.

<- error_event

Called for system errors

Class WordsClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [WordsClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class WordsClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [WordsClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class WordsClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [WordsClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class WordsClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [WordsClass Methods](#) for more information.

<- error_event

Called for system errors

Class SpreadsheetsClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [SpreadsheetsClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class SpreadsheetsClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [SpreadsheetsClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class SpreadsheetsClass event

Format this.terminate_event

Description The terminate_event is called by Appixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [SpreadsheetsClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class SpreadsheetsClass event

Format this.time_out_event

Description The time_out_event is called by Appixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [SpreadsheetsClass Methods](#) for more information.

-> document_exit_event

Called when exiting a document

Class event

Format this.document_exit_event(file)

Arguments file The name of the document file.

Description The document_exit_event is called before closing the document. This is a user-defined event, place all actions you want performed in the event definition. For example, you may want the application to clean up after exiting a file or to signal a different part of the application.

-> document_modified_event

Called when saving a document

Class event

Format this.document_modified_event(file)

Arguments file The name of the document file.

Description The document_modified_event is called when saving the current document. This is a user-defined event, place all actions you want performed in the event definition. For example, you may want the application to verify the file name when saving the document.

-> document_open_event

Called when opening a document

Class event

Format this.document_open_event(file)

Arguments file The name of the document file.

Description The document_open_event is called when opening a document. This is a user-defined event, place all actions you want performed in the event definition. For example, you may want the application to perform a permission check before opening the file.

-> task_exit_event

Called when exiting an application window

Class event

Format this.task_exit_event

Description The `task_exit_event` is called before closing the application window. This is a user-defined event, place all actions you want performed in the event definition. For example, you may want the application to clean up after exiting an application or to signal a different part of the application.

BUILDER_APPLICATION@

Runs an application in a new task

Format `object app = BUILDER_APPLICATION@(file[, arg1[, arg2[, arg3[, arg4[, arg5]]]])`

Arguments

<code>file</code>	The Applixware Builder file, with full file path. If <code>file</code> is passed as an array, <code>file[0]</code> is used as the file name, and all other array items are taken as application arguments.
<code>arg1</code>	An optional application argument.
<code>arg2</code>	An optional application argument.
<code>arg3</code>	An optional application argument.
<code>arg4</code>	An optional application argument.
<code>arg5</code>	An optional application argument.

Description Runs an Applixware Builder application in a new task and returns the application object. Use the macro to run multiple applications against the same `axmain` process. The data types of the optional arguments are defined by the target application.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

See also [PENDING FOR BUILDER_APPLICATION@](#)

BUILDER_APPLICATION_DLG@

Runs Applixware Builder in a new task

Format `BUILDER_APPLICATION_DLG@([file[, arg1]])`

Arguments file The Applixware Builder file, with full file path. If file is passed as an array, file[0] is used as the file name, and all other array items are taken as application arguments. If no file is passed, a new file is opened in Applixware Builder.

arg1 An optional application argument.

Description Runs Applixware Builder in a new task. Use the macro to run multiple Applixware Builder processes against the same axmain process. The data types of the optional arguments are defined by the target application.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

See also [BUILDER_APPLICATION@](#)

BUILDER_APPLICATIONS_IN_MEMORY@

Returns the application objects in memory

Format var object apps = BUILDER_APPLICATIONS_IN_MEMORY@([name])

Arguments name The optional name of the ApplicationClass object.

Description The macro returns the ApplicationClass objects that exist in memory and have the passed name. If no name is supplied, all ApplicationClass objects are returned.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

BUILDER_COMPILE@

Compiles an application file into an output file

Format BUILDER_COMPILE@(inputFile, outputFile, localFlag[,macroFile[, commandFile[, searchList]]])

Arguments inputFile The Applixware Builder application (.ab) file name, with full file path.

outputFile The output executable (.abo) file name, with full path.

localFlag A Boolean value, TRUE means localize all external classes, objects, and resources in the file before creating the output file. The default value is FALSE.

- macroFile An optional argument, the name of the macro to execute the output file. This is the same as using Tools ® Make Macro to create an application execution macro.
- commandFile An optional argument, the name of the UNIX shell script file name. Use this argument to create a shell script which you can use to run the executable against elfrt.
- searchList An optional argument, an array of directories to add to the ELF search list. This argument should contain the directories for externally referenced files.

Description The macro compiles the passed application file into an executable output file. Use the optional arguments to create a macro or shell script you can use to run the executable. Use this macro with elfrt in a shell script or make file to create an executable without running Applixware Builder.

See also [BUILDER_CREATE_DISTRIBUTION_FILE@](#)
[BUILDER_UNPACK_DISTRIBUTION_FILE@](#)

BUILDER_CREATE_DISTRIBUTION_FILE@

Creates a distribution file

Format BUILDER_CREATE_DISTRIBUTION_FILE@(inputFile, outputFile, turboFlag, localFlag)

- Arguments**
- inputFile The Applixware Builder application (.ab) file name, with full file path.
 - outputFile The output distribution (.abd) file name, with full path.
 - turboFlag A Boolean value, TRUE means create an executable output (.abo) file in the distribution file. FALSE means place the application (.ab) file in the distribution file. The default value is FALSE.
 - localFlag A Boolean value, TRUE means the reference paths for all external classes, objects, and resources in the application file will be changed to ./, so that the unpacking the distribution file places all external files in the same directory as the application file. The default value is FALSE.

Description The macro creates a distribution file for the passed application file. Use this macro with elfrt in a shell script or make file to create a distribution files without running Applixware Builder.

See also [BUILDER_COMPILE@](#)

BUILDER_UNPACK_DISTRIBUTION_FILE@

BUILDER_INSTALL_DISTRIBUTION@

Installs a distribution file with the Install Builder Distribution dialog box

Format BUILDER_INSTALL_DISTRIBUTION@(name)

Arguments name The full path name of the Applixware Builder distribution file, including .abd file extension.

Description The macro installs an Applixware Builder distribution file with the Install Builder Distribution dialog box. You can use the dialog box to interactively install all the files within the distribution file.

BUILDER_READ_DOC@

Reads and returns an application file

Format format builder_docinfo@ doc = BUILDER_READ_DOC@(name, objFlag,turboFlag)

Arguments name The full path name of the Applixware Builder application file, including .ab file extension.

objFlag A Boolean value, TRUE means return only the ApplicationClass object, FALSE means return the entire application in builder_docinfo@ format.

turboFlag A Boolean value, TRUE means read the file and compile the application for execution.

Description The macro reads and returns an Applixware Builder application file in builder_docinfo@ format. The format is defined in the builder_.am header file, located in the /install_dir/axdata/elf directory. If the objFlag is set to TRUE then only the ApplicationClass object is returned.

BUILDER_RUNTIME_DLG@

Runs the Applixware Builder Run resource

Format BUILDER_RUNTIME_DLG@()

Description Runs the Applixware Builder Run resource. Use the resource to run multiple applications and macros, or to install applications in distribution format. The Applixware Builder

Run dialog box appears when you run the macro. The dialog box is the interface to the resource.

BUILDER_UNPACK_DISTRIBUTION_FILE@

Unpacks a distribution file

Format BUILDER_UNPACK_DISTRIBUTION_FILE@(inputFile, outputDir)

Arguments

inputFile	The Applixware Builder distribution (.abd) file name, with full file path.
outputDir	The output directory for unpacking the application, and the reference for installing supporting files with relative path names.

Description The macro unpacks a distribution file in the passed output directory.

Use this macro with elfrt in a shell script or make file to unpack distribution files without running Applixware Builder.

See also [BUILDER_COMPILE@](#)
[BUILDER_CREATE_DISTRIBUTION_FILE@](#)

BUILDER_WRITE_DOC@

Writes an Applixware Builder application to file

Format BUILDER_WRITE_DOC@(name, format builder_docinfo@ doc ,turboFlag)

Arguments

name	The full path name of the Applixware Builder application file, including .ab file extension.
doc	The Applixware Builder application file, in builder_docinfo@ format. The format is defined in the builder_.am header file, located in the <i>/install_dir/axdata/elf</i> directory.
turboFlag	A Boolean value, TRUE means write the application to file as a turbo (.abo) executable file. FALSE means write the application to file as an application (.ab) source file.

Description The macro writes the Applixware Builder application to file.

BUILDER_WRITE_TURBO_DOC@

Writes an Applixware Builder application to file in turbo format

Format BUILDER_WRITE_TURBO_DOC@(name, format builder_docinfo@ doc = ,turboFlag)

Arguments

name	The full path name of the Applixware Builder application file, including .ab file extension.
doc	The Applixware Builder application file, in builder_docinfo@ format. The format is defined in the builder_.am header file, located in the <i>install_dir/axdata/elf</i> directory.

Description The macro writes the Applixware Builder application to file in turbo format.

IS_APPLICATION_RUNNING@

Returns application status

Format flag = IS_APPLICATION_RUNNING@(object app)

Arguments app An Applixware Builder application object.

Description Returns application status as a Boolean value. The macro returns TRUE if the passed application object is running, otherwise the macro returns FALSE.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

See also [BUILDER_APPLICATION@](#)
[PEND_FOR_BUILDER_APPLICATION@](#)
[PROMOTE_APPLICATION@](#)

PEND_FOR_BUILDER_APPLICATION@

Runs an application as a pended task

Format data = PEND_FOR_BUILDER_APPLICATION@(file[, arg1[, arg2[, arg3[, arg4[, arg5]]]])

Arguments file The Applixware Builder file, with full file path. If file is passed as an array, file[0] is used as the file name, and all other array items are taken as application arguments.

arg1 An optional application argument.
arg2 An optional application argument.
arg3 An optional application argument.
arg4 An optional application argument.

Description Runs an Applixware Builder application as a pended task. The application from which you call the macro waits for a return signal from the pended task. The data types of the optional arguments are defined by the target application.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

See also [BUILDER_APPLICATION@](#)

PROMOTE_APPLICATION@

Promotes application main dialog box

Format flag = PROMOTE_APPLICATION@(object app)

Arguments app An Applixware Builder application object.

Description Promotes the application object main dialog box to the front of the screen. If the application is running, the macro promotes the application's main dialog box and returns TRUE. If the application is not running, or if the application main dialog has been displayed, the macro returns FALSE.

Use this macro with [ApplicationClass](#) methods to manipulate application objects.

See also [BUILDER_APPLICATION@](#)
[IS_APPLICATION_RUNNING@](#)
[PEND_FOR_BUILDER_APPLICATION@](#)

<- all_data_sets@

Returns all data sets in the application

Class ApplicationClass get

Format objectArray = this.all_data_sets@

Description The get method `all_data_sets@` returns all data set objects in the application as an array.

For example, to get all data sets in the application use the following:

```
var object objectArray
objectArray = this.all_data_sets@
```

See [ApplicationClass Methods](#) for more information.

`<- all_dboxes@`

Returns all dialog boxes in the application

Class ApplicationClass get

Format objectArray = this.all_dboxes@

Description The get method `all_dboxes@` returns all dialog boxes in the application as an array.

For example, to get all dialog boxes in the application use the following:

```
var object objectArray
objectArray = this.all_dboxes@
```

See [ApplicationClass Methods](#) for more information.

`<- all_printers@`

Returns all PrinterClass objects in the application

Class ApplicationClass get

Format objectArray = this.all_printers@

Description The get method `all_printers@` returns all PrinterClass objects in the application as an array.

For example, to get all PrinterClass objects in the application use the following:

```
var object objectArray
objectArray = this.all_printers@
```

See [ApplicationClass Methods](#) for more information.

<- all_rt_gateways@

Returns all Real Time gateways in the application

Class ApplicationClass get

Format objectArray = this.all_rt_gateways@

Description The get method all_rt_gateways@ returns all Real Time gateways in the application as an array.

For example, to get all Real Time gateways in the application use the following:

```
var object objectArray
objectArray = this.all_rt_gateways@
```

See [ApplicationClass Methods](#) for more information.

<- arg_count@

Returns the quantity of arguments passed to the application

Class ApplicationClass get

Format number = this.arg_count@

Description The get method arg_count@ returns the quantity of arguments passed to the application.

For example, to get the quantity of arguments passed to the application use the following:

```
var number
number = this.arg_count@
```

See [ApplicationClass Methods](#) for more information.

<- arg_var@

Returns an application argument

Class ApplicationClass get

Format `arg = this.arg_var@(index)`

Arguments `index` The index in the array of arguments.

Description The get method `arg_var@` returns an argument from the array of arguments passed to the application. Use the get method `arg_vars@` to retrieve the argument array.

For example, to get argument 0 use the following:

```
var argument
argument = this.arg_var@(0)
```

See [ApplicationClass Methods](#) for more information.

<- arg_vars@

Returns an array of application argument

Class `ApplicationClass` `get`

Format `arguments = this.arg_vars@`

Description The get method `arg_vars@` returns the array of arguments passed to the application.

For example, to get all arguments passed to the application use the following:

```
var arguments
arguments = this.arg_vars@
```

See [ApplicationClass Methods](#) for more information.

<- data_set_find@

Returns a data set by name

Class `ApplicationClass` `get`

Format `dataObj = this.data_set_find@(name)`

Arguments `name` The data set name to find.

Description The get method `data_set_find@` returns the data set with a name matching the passed string. The method returns NULL if the data set does not exist.

For example, to get the data set `Data1` use the following:

```
var object data
data = this.data_set_find@("Data1")
```

See [ApplicationClass Methods](#) for more information.

<- data_source_find@

Returns a data source by name

Class ApplicationClass get

Format dataObj = this.data_source_find@(name)

Arguments name The data source name to find.

Description The get method data_source_find@ returns the data source with a name matching the passed string as an object. The method searches for data sets and Real Time data sources. The method returns NULL if the data source does not exist.

For example, to get the data source Data1 use the following:

```
var object data
data = this.data_set_find@("Data1")
```

See [ApplicationClass Methods](#) for more information.

<- dbox_find@

Returns a dialog box object

Class ApplicationClass get

Format object = this.dbox_find@(name)

Arguments name A string for the dialog box name, or the object.

Description The get method dbox_find@ returns a dialog box object given the name or the actual object.(If the named dialog box does not exist, the dialog box object is created as a child of the object.)

For example, to get a dialog box named form:Example object use the following:

```
var object dbox
dbox = this.dbox_find@("Example")
```

The form: part of the dialog box name is optional.

See [ApplicationClass Methods](#) for more information.

<- dbox_is_open@

Returns dialog box open status

Class ApplicationClass get

Format flag = this.dbox_is_open@(name)

Arguments name A string for the dialog box name, or the object.

Description The get method dbox_is_open@ returns the open status of the dialog box as a Boolean value. The method returns TRUE if the dialog box is open, otherwise it returns FALSE.

For example, to get the open status of dialog box form:Example use the following:

```
var flag
flag = this.dbox_is_open@("form:Example")
```

The form: part of the dialog box name is optional.

See [ApplicationClass Methods](#) for more information.

<- dbox_main@

Returns the main application dialog box

Class ApplicationClass get

Format object = this.dbox_main@

Description The get method dbox_main@ returns the main application dialog box. You can use the returned object to open or close the main dialog box, or perform other actions.

For example, to get the main application dialog box use the following:

```
var object formObject
formObject = this.dbox_main@
```

See [ApplicationClass Methods](#) for more information.

<- `dbox_open@`

Opens a response dialog box

Class ApplicationClass get

Format number = this.dbox_open@(name[,arg1[,arg2[,arg3[,arg4[,arg5]]]])

Arguments name A string for the dialog box name, or the object.

1. .arg5 Optional arguments.

Description The get method `dbox_open@` opens a response dialog box or promotes a response dialog box if it is already open. The method returns a user value from the response dialog box.

For example, to open the response dialog box form:Example you would use the following:

```
var value
value = this.dbox_open@("form:Example")
```

See [ApplicationClass Methods](#) for more information.

<- `error_object@`

Returns an error object

Class ApplicationClass get

Format arg = this.error_object@

Description The get method `error_object@` returns an error object. Use the method to retrieve error information when an application error occurs.

For example, to get an error object use the following:

```
var object error
error = this.error_object@
```

See [ApplicationClass Methods](#) for more information.

<- remote_object_create@

Creates a remote object

Class ApplicationClass get

Format obj = this.remote_object_create@(host, class, name)

Arguments

host	The name of the host machine where the object exists.
class	The name of an object class in the current application from which the object inherits attributes and methods. An object class can be a standard Applixware Builder class, or a user-defined class.
name	The object name, as a text string.

Description The method creates and returns a remote object. An axnet process must exist on the host machine before you create a remote object with this macro. A remote object is referenced by the returned object handle, the object cannot be assigned as the child of a local application object. A remote object exists until it is removed with the set method [remote_object_destroy@](#).

The axnet process on the host machine must be started using the full path name. If axnet is started without the full path name, or with a linked path name, the method is unable to find the process and create a remote object.

See [ApplicationClass Methods](#) for more information.

<- remote_object_find@

Returns a remote object

Class ApplicationClass get

Format obj = this.remote_object_find@(host, name)

Arguments

host	The name of the host machine where the object exists.
name	The name of the object to find.

Description The method returns a handle to a remote object. The method returns NULL if the the object does not exist on the remote host.

<- rt_gateway_find@

Returns a Real Time gateway object

Class ApplicationClass get

Format object = this.rt_gateway_find@(name)

Arguments name A string for the Real Time gateway name, or the object.

Description The get method `rt_gateway_find@` returns a Real Time gateway object given the name or the actual object. If the named Real Time gateway does not exist, the method returns NULL.

For example, to get a Real Time gateway named `data:RealTime` object use the following:

```
var object gate
gate = this.rt_gateway_find@("data:RealTime")
```

See [ApplicationClass Methods](#) for more information.

<- send_message_get_response@

Sends message to application, returns the message sent back

Class ApplicationClass get

Format returnMsg = this.send_message_get_response@(object app, code, data)

Arguments app The application object receiving the message.

code The message code.

data The message data.

Description The get method `send_message_get_response@` sends a message to an application, then returns the reply message. If the passed application object is not running, no message is sent. A message sent with this method is handled by the receiving application's [message respond event](#). The application calling this method pends for the response from the application receiving the message.

See [ApplicationClass Methods](#) for more information.

<- task_id@

Returns application task ID

Class ApplicationClass get

Format id = this.task_id@

Description The get method task_id@ returns the application's task ID. Each application has a unique task identification number.

For example, to get the application's task ID use the following:

```
var id
id = this.task_id@
```

See [ApplicationClass Methods](#) for more information.

-> arg_var@

Sets an argument value

Class ApplicationClass set

Format this.arg_var@(index, value)

Arguments

index	The index in the array of arguments.
value	The new value for the argument.

Description The set method arg_var@ changes an argument passed to the application. Use this method with the get method arg_var@ to check and change arguments.

For example, to set argument 1 to Blue use the following:

```
var argument
argument = this.arg_var@(1) ' get method
IF argument <> "Blue"
    this.arg_var@(1, "Blue") ' set method
```

See [ApplicationClass Methods](#) for more information.

-> default_error_handler@

Sets system default error handler

Class ApplicationClass set

Format this.default_error_handler@

Description The set method default_error_handler@ sets the system default error handler. The system error handler is used by default to handle error messages. If an **error event** exists, system errors are handled by the error_event. Use default_error_handler@ to pass error handling to the system after you handle it in your error_event code.

See [ApplicationClass Methods](#) for more information.

-> dbox_close@

Closes a dialog box

Class ApplicationClass set

Format this.dbox_close@(name)

Arguments name A string for the dialog box name, or the object.

Description The set method dbox_close@ closes a dialog box.

For example, to close the dialog box form:Example use the following:

```
    this.dbox_close__("form:Example")
```

See [ApplicationClass Methods](#) for more information.

-> dbox_disable@

Disables a dialog box

Class ApplicationClass set

Format this.dbox_disable@(name)

Arguments name A string for the dialog box name, or the object.

Description The set method `dbox_disable@` disables a dialog box. All widgets in the dialog box are grayed and become inactive. Use this method to make dialog box actions unavailable without closing the dialog box.

For example, to disable the dialog box form:Example use the following:

```
this.dbox_disable@("form:Example")
```

See [ApplicationClass Methods](#) for more information.

-> `dbox_enable@`

Enables a dialog box

Class ApplicationClass set

Format `this.dbox_enable@(name)`

Arguments name A string for the dialog box name, or the object.

Description The set method `dbox_enable@` enables a dialog box. All widgets in the dialog box are ungrayed and initialized.

For example, to enable the dialog box form:Example use the following:

```
this.dbox_enable@("form:Example")
```

See [ApplicationClass Methods](#) for more information.

-> `dbox_open@`

Opens a dialog box

Class ApplicationClass set

Format `this.dbox_open@(name)`

Arguments name A string for the dialog box name, or the object.

Description The set method `dbox_open@` opens a dialog box or promotes a dialog box if it is already open.

For example, to open the response box form:Example use the following:

```
this.dbox_open@("form:Example")
```

See [ApplicationClass Methods](#) for more information.

-> `dbox_update@`

Refreshes a dialog box

Class ApplicationClass set

Format `this.dbox_update@(name)`

Arguments name A string for the dialog box name, or the object.

Description The set method `dbox_update@` refreshes a dialog box display, redrawing all the information within the dialog box.

For example, to refresh the dialog box form:Example use the following:

```
this.dbox_update@(form:Example)
```

See [ApplicationClass Methods](#) for more information.

-> `is_windowless@`

Sets windowless state of application

Class ApplicationClass set

Format `this.is_windowless@(flag)`

Arguments flag A Boolean value, TRUE runs an application windowlessly.

Description The set method `is_windowless@` sets the windowless state of the application. You need to use this method in the ApplicationClass object `initialize_event` for the application to run in windowless mode.

You need to use this method carefully. If you are not sure if the a windowless application will terminate properly you will need to get the task ID of the windowless application. You can use the task ID to kill the task, otherwise the application may continue execution.

See [ApplicationClass Methods](#) for more information.

-> object_server_close@

Closes connection to remote object server

Class ApplicationClass set

Format this.object_server_close@(host)

Arguments host The name of the host machine.

Description The set method object_server_close@ closes an axnet connection to the host object server machine. Use this method to close the connection to a remote server after you delete all remote objects with [remote object destroy@](#).

See [ApplicationClass Methods](#) for more information.

-> object_server_open@

Opens connection to remote object server

Class ApplicationClass set

Format this.object_server_open@(host)

Arguments host The name of the host machine.

Description The set method object_server_open@ opens an axnet connection to the host object server machine. Use this method to open the connection to a remote machine before you create remote objects with [remote object create@](#).

See [ApplicationClass Methods](#) for more information.

-> quit@

Exits an application

Class ApplicationClass set

Format this.quit@

Description The set method quit@ exits an application, closing all dialog boxes and terminating all processes.

For example, to exit an application use the following:

```
this.quit@
```

See [ApplicationClass Methods](#) for more information.

-> remote_object_destroy@

Removes a remote object

Class ApplicationClass set

Format this.remote_object_destroy@(host, object obj)

Arguments host The name of the host machine where the object exists.
obj The object to delete.

Description The set method remote_object_destroy@ removes the remote object from the host machine. The method does not close the axnet connection to the remote host. Use the set method [object_server_close@](#) method in the CommonDlgClass to close the axnet connection.

See [ApplicationClass Methods](#) for more information.

-> return_value@

Sets the return value of an application in pend mode

Class ApplicationClass set

Format this.return_value@(value)

Arguments value A return value of any variable type.

Description The set method return_value@ sets the value an application in pend mode returns. For example, to set the return value to the object exampleObject use the following:

```
this.return_value@(exampleObject)
```

See [ApplicationClass Methods](#) for more information.

-> send_message@

Sends message to application

Class ApplicationClass set

Format this.send_message@(object app, code, data)

Arguments

app	The application object receiving the message.
code	The message code.
data	The message data.

Description The set method send_message@ sends a message to an application. If the passed application object is not running, no message is sent. A message sent with this method is handled by the receiving application's [message event](#).

See [ApplicationClass Methods](#) for more information.

-> task_id@

Sets application task ID

Class ApplicationClass set

Format this.task_id@(id)

Arguments id A task identification number.

Description The set method task_id@ sets the application's task ID. Each application has a unique task identification number.

For example, to set the application's task ID use the following:

```
    this.task_id@(500)
```

See [ApplicationClass Methods](#) for more information.

<- error_event

Called for system errors

Class ApplicationClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting system errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ApplicationClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class ApplicationClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting the application main dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ApplicationClass Methods](#) for more information.

-> message_event

Called when application receives a message

Class ApplicationClass event

Format this.message_event(code, data)

Arguments code The message code.
 data The information passed.

Description The message_event is called by Applixware Builder when the application receives an asynchronous message sent from another application with the [send_message@](#) or method. This is a user-defined event, place all actions you want performed in the event definition.

See [ApplicationClass Methods](#) for more information.

<- message_respond_event

Called when application receives a response

Class ApplicationClass event

Format value = this.message_respond_event(code,data)

Arguments code The message code.
 data The information passed.

Description The message_respond_event is called by Applixware Builder when the application receives an synchronous message sent from another application with the [send_message_get_response@](#) method. This is a user-defined event, place all actions you want performed in the event definition. The event should return the appropriate response value the calling application is expecting..

If an error occurs in the receiving application, and the application handles the error, rethrow the error after the application handles the error. The application sending the synchronous message receives the error after it is rethrown.

See [ApplicationClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class ApplicationClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [ApplicationClass Methods](#) for more information.

-> `time_out_event`

Called on object timer time-out

Class `ApplicationClass` event

Format `this.time_out_event`

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [ApplicationClass Methods](#) for more information.

<- `all_children@`

Returns all child objects

Class `BaseClass` get

Format `objectArray = this.all_children@`

Description The get method `all_children@` returns all child objects, including children of immediate child objects. Use the method to find all objects that descend from the current object.

For example, to get all children use the following:

```
var object objectArray
objectArray = this.all_children@
```

See [BaseClass Methods](#) for more information.

<- application@

Returns top-level application object

Class BaseClass get

Format app = this.application@

Description The get method application@ returns the top-level application object. Use the top-level application object to perform operations not accessible with the current object or parent object, such as closing dialog boxes or exiting an application.

For example, to get the application use the following:

```
var object app
app = this.application@
```

See [BaseClass Methods](#) for more information.

<- application_name@

Returns application object name

Class BaseClass get

Format appName = this.application_name@

Description The get method application_name@ returns the application object name as a string.

For example, to get the application object name use the following:

```
var appName
appName = this.application_name@
```

See [BaseClass Methods](#) for more information.

<- child@

Returns a child object for a passed text string

Class BaseClass get

Format object = this.child@(name)

Arguments name A string for the child object name.

Description The get method child@ returns a child object with the same name as a passed string. The method throws an error if the child object does not exist. Use the get method is_child@ to verify the child object exists.

For example, to get a child named Button use the following:

```
var object childObject
childObject = this.child@("Button")
```

See [BaseClass Methods](#) for more information.

<- children@

Returns an array of child objects

Class BaseClass get

Format objectArray = this.children@

Description The get method children@ returns an array of immediate child objects, an empty array if the object has no children.

For example, to get the child objects of the current object use the following:

```
var object childObjects
childObjects = this.children@
```

See [BaseClass Methods](#) for more information.

<- class_library@

Returns a class object reference

Class BaseClass get

Format classObjectRef = this.class_library@(libName)

Arguments libName The name of the .dll file that you loaded into Builder that defines the class.

Description The class_library@ method is used when you create an object from a user-defined class. The user-defined class is loaded into Builder from a .dll file.

The classObjectRef value returned by this method is an object reference to a class object. You should pass this class object reference to the macro OBJECT_CREATE@ to instantiate a new builder object.

See also [OBJECT_CREATE@](#)

<- data_set_find@

Returns an object reference to a named dataset object

Class BaseClass get

Format object var = this.data_set_find@(objectName)

Arguments objectName The name of a dataset object

Description Returns an object reference to the named DatasetClass object. This object must be part of the Builder application, otherwise the method returns a NULL value.

<- comments@

Returns application comments

Class BaseClass get

Format stringArray = this.comments@

Description The get method comments@ returns the comments stored with an application as an array of strings.

For example, to get the application comments use the following:

```
var stringArray
stringArray = this.comments@
```

See [BaseClass Methods](#) for more information.

<- inherit@

Returns the inheritance class

Class BaseClass get

Format object = this.inherit@

Description The get method inherit@ returns the inheritance class. The inheritance class defines the methods and events available for the current object. The current object is an instance of the inheritance class.

For example, to get the inheritance class of the current object use the following:

```
var object classObject
classObject = this.inherit@
```

See [BaseClass Methods](#) for more information.

<- is_child@

Returns child status

Class BaseClass get

Format flag = this.is_child@(name)

Arguments name A string for the child object name.

Description The get method is_child@ returns child status as a Boolean value. The method returns TRUE if the object is a child of the current object. The method returns FALSE if the object is not a child of the current object or does not exist.

For example, to get a child status use the following:

```
var flag
flag = this.is_child@("Button")
```

See [BaseClass Methods](#) for more information.

<- load_methods_file@

Sets the methods to a source file

Class BaseClass set

Format retVal = obj.load_methods_file@(fileName[,dumpError])

Arguments fileName A string for the full pathname of the source file, or an array of strings containing the method source.

`dumpError` An optional Boolean argument, TRUE means dump the compilation errors into the Array List dialog box and display the dialog box. FALSE means return compilation errors in a two-dimensional array. The default value is FALSE.

`retVal[0]` contains the object's hierarchy, beginning with the object, then its parent and all other objects that it descends from. `retVal[1]` contains the array of error strings.

Description The get method `load_methods_file@` sets the available methods for an object to the methods contained in a passed source file and returns any compilation errors. If no compilation errors occur, or if you dump the errors, the methods returns NULL. The source file contains ASCII text defining methods. Use this method if want want to handle compilation errors within your application code.

For example, to set the current object methods to the methods defined in the source file `/user/Applixware/build_method` use the following:

```
var object button
var retVal
button = this.child@("Button")
retVal = button.load_methods_file@("/user/Applixware/build_method")
IF retVal <> NULL
{...
    code to handle a compilation error
...}
```

To set the current object methods with an array of strings use the following:

```
var object button
button = this.child@("Button")
retVal = this.load_methods_file@ = {
    "@@@OBJECTS",
    "set initialize_event",
    "endset"
}
```

See also set [load_methods_file@](#)

See [BaseClass Methods](#) for more information.

<- name@

Returns the name of the object as a text string

Class BaseClass get

Format string = this.name@

Description The get method name@ returns the name of the object.

For example, to retrieve the name of the current object as a string variable use the following:

```
var string
string = this.name@
```

If the object's name is Button2, the string variable contains the text string Button2.

See [BaseClass Methods](#) for more information.

<- parent@

Returns the parent object

Class BaseClass get

Format object = this.parent@

Description The get method parent@ returns the parent object. Use the parent object to perform operations not accessible with the current object.

For example, to get the parent object of the current object use the following:

```
var object parentObject
parentObject = this.parent@
```

See [BaseClass Methods](#) for more information.

<- public_objvars@

Returns public objvars

Class BaseClass get

Format vars = this.public_objvars@

Description The get method public_objvars@ returns the public objvars of an object. Public objvars are object variables you can reference from other objects without using methods. Use the set method public_objvars@ to define the variables you can set or get without methods.

For example, to get the public objvars of an object use the following:

```
var variables
variables = this.public_objvars@
```

See [BaseClass Methods](#) for more information.

<- quick_help@

Returns quick help string for a method

Class BaseClass get

Format helpStr = this.quick_help@(type, name)

Arguments type The type of method: set or get.
name A method name in the class.

Description The get method quick_help@ returns the quick help string for a method in the class. Use this method with user-defined classes to determine if a quick help string is assigned to the method. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name.

See also set [quick_help@](#)
set [quick_help_initialize_event](#)

See [BaseClass Methods](#) for more information.

<- resource_content@

Returns contents of a resource file

Class BaseClass get

Format info = this.resource_content@(file)

Arguments file The resource file path name.

Description The get method `resource_content@` returns the contents of a resource file. Resource files are loaded into an application with the **Load Resources** dialog box. The method returns NULL if the resource file does not exist or if it is not loaded as a resource.

Resource information is saved as part of the top-level ApplicationClass object. Use this method with the ApplicationClass object to get the contents of a resource file.

See **BaseClass Methods** for more information.

<- sibling@

Returns an object sharing the same parent

Class BaseClass get

Format object = this.sibling@(name)

Arguments name A string for the sibling object name.

Description The get method `sibling@` returns an object sharing the same parent as the current object. The method throws an error if the sibling does not exist.

For example, to get a sibling named `brotherButton` use the following:

```
var object siblingObject
siblingObject = this.sibling@("brotherButton")
```

See **BaseClass Methods** for more information.

<- source_strings@

Returns an object's source

Class BaseClass get

Format stringArray = this.source_strings@

Description The get method `source_strings@` returns an object's source as an array of strings. Use the method to determine the methods and events defined for the object.

For example, to get an object's source use the following:

```
var stringArray
```

```
stringArray = this.source_strings@
```

See [BaseClass Methods](#) for more information.

<- timer@

Returns an object's timer setting

Class BaseClass get

Format time = this.timer@

Description The get method timer@ returns an object's timer setting, in seconds. The timer value determines the frequency of when a [time out event](#) is generated. Use the set method [timer@](#) to set the time-out interval.

See [BaseClass Methods](#) for more information.

-> add_child@

Adds an object as a child

Class BaseClass set

Format this.add_child@(object)

Arguments object An object created with the [OBJECT CREATE@](#) macro.

Description The set method add_child@ adds an object to the current object's list of children. For example, to add an object buttonExample to the current object use the following:

```
this.add_child@(buttonExample)
```

See [BaseClass Methods](#) for more information.

>- application_name@

Sets application object name

Class BaseClass set

Format this.application_name@(name)

Arguments name A string for the application object name.

Description The set method `application_name@` sets the application object name to the passed string.

For example, to set the application object name use the following:

```
var appName
this.application_name@("Application")
```

See [BaseClass Methods](#) for more information.

-> **comments@**

Sets application comments

Class BaseClass set

Format `this.comments@(stringArray)`

Arguments stringArray A single string, or array of strings

Description The set method `comments@` sets the comments stored with an application to the passed string or array of strings.

For example, to set the application comments use the following:

```
var stringArray
stringArray[0] = "First line of comments"
stringArray[1] = "Last line of comments"
this.comments@(stringArray)
```

See [BaseClass Methods](#) for more information.

-> **delete_child@**

Removes a child object

Class BaseClass set

Format `this.delete_child@(objectName)`

Arguments objectName The child object to delete.

Description The set method `delete_child@` removes the passed child object from the current object. Use the get method `child@` to get a pointer to the object, then pass the returned object to `delete_child@`.

For example, to remove the child object `exampleButton` use the following:

```
var object childObject
childObject = this.child@("exampleButton")
this.delete_child@(childObject)
```

See [BaseClass Methods](#) for more information.

-> `delete_children@`

Removes all child objects

Class BaseClass set

Format `this.delete_children@`

Description The set method `delete_children@` removes all child objects from the current object.

For example, to remove all child objects from the current object use the following:

```
this.delete_children@
```

See [BaseClass Methods](#) for more information.

-> `inherit@`

Changes the inherit class

Class BaseClass set

Format `this.inherit@(className)`

Arguments `className` A string for the class name.

Description The set method `inherit@` changes the inherit class of the current object to the passed class name.

For example, to change the inherit class of the current object to `ButtonClass` use the following:

```
this.inherit@("ButtonClass")
```

or

```
this.inherit@ = "ButtonClass"
```

See [BaseClass Methods](#) for more information.

-> load_methods_file@

Sets the methods to a source file

Class BaseClass set

Format obj.load_methods_file@(fileName)

Arguments fileName A string for the full pathname of the source file, or an array of strings.

Description The set method load_methods_file@ sets the available methods for an object to the methods contained in a passed source file. The source file contains ASCII text defining methods.

For example, to set the current object methods to the methods defined in the source file /user/Applixware/build_method use the following:

```
var object button
button = this.child@("Dismiss")
button.load_methods_file@("/user/Applixware/build_method")
```

To set the current object methods with an array of strings use the following:

```
var object button
button = this.child@("Dismiss")
button.load_methods_file@ = {
    "@@@OBJECTS",
    "set initialize_event",
    "endset"
}
```

See also get [load_methods_file@](#)

See [BaseClass Methods](#) for more information.

-> name@

Sets the name of the object

Class BaseClass set

Format this.name@(string)

Arguments string A string for the object name.

Description The set method name@ sets the name of the object to the passed string.

For example, to set the name of the current object to CancelButton use the following:

```
    this.name@ = "CancelButton"
```

See [BaseClass Methods](#) for more information.

-> parent@

Sets the parent object

Class BaseClass get

Format this.parent@(object)

Arguments object An object in the application.

Description The set method parent@ sets the parent object of the current object.

For example, to set the parent object of the current object use the following:

```
    var object parentObject
    parentObject = this.application@.child@("Form")
    this.parent@(parentObject)
```

See [BaseClass Methods](#) for more information.

-> public_objvars@

Sets public objvars

Class BaseClass set

Format `this.public_objvars@(nameArray)`

Arguments `nameArray` An array of object names.

Description The set method `public_objvars@` sets the public objvars. Public objvars are object variables you can reference from other objects without using methods. Use the method to define the variables you can set or get without methods.

For example, to define object variables as public objvars, use the following:

```
objvar objvar1
```

```
objvar objvar2
```

```
this.public_objvars@ = {"objvar1", "objvar2"}
```

See [BaseClass Methods](#) for more information.

-> `quick_help@`

Sets quick help string for a method

Class BaseClass set

Format `this.quick_help@(type, name, helpStr)`

Arguments

<code>type</code>	The type of method: set or get.
<code>name</code>	A method name in the class.
<code>helpStr</code>	The quick help string

Description The set method `quick_help@` sets the quick help string for a method in the class. Use this method in the [quick_help_initialize_event](#) of user-defined classes to assign a quick help string to a method. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name.

See also get [quick_help@](#)

See [BaseClass Methods](#) for more information.

-> `timer@`

Sets object time-out timer

Class BaseClass set

Format this.timer@(time)

Arguments time A time value, in seconds.

Description The set method timer@ sets the object's time-out timer. Use the timer to generate a **time_out_event**. A time_out_event is generated when the object's timer expires.

For example, a clock application would use this method to set the timer to 1 second. A time_out_event would occur after 1 second, then the code in the time_out_event would set the new time and display it.

See **BaseClass Methods** for more information.

<- error_event

Called for system errors

Class BaseClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use **ErrorClass** methods to get error object information.

See **BaseClass Methods** for more information.

-> initialize_event

Called before posting application dialog box

Class BaseClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [BaseClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class BaseClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [BaseClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class BaseClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [BaseClass Methods](#) for more information.

-> quick_help_initialize_event

Initializes quick help strings for class methods

Class BaseClass event

Format this.quick_help_initialize_event

Description The quick_help_initialize_event initializes the quick help strings for methods in a user-defined class. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name. Use the set method [quick_help@](#) in this event to define the quick help strings for methods in the class.

See also get [quick_help@](#)

See [BaseClass Methods](#) for more information.

<- all_children@

Returns all child objects

Class BaseClass get

Format objectArray = this.all_children@

Description The get method all_children@ returns all child objects, including children of immediate child objects. Use the method to find all objects that descend from the current object.

For example, to get all children use the following:

```
var object objectArray
objectArray = this.all_children@
```

See [BaseClass Methods](#) for more information.

<- application@

Returns top-level application object

Class BaseClass get

Format app = this.application@

Description The get method application@ returns the top-level application object. Use the top-level application object to perform operations not accessible with the current object or parent object, such as closing dialog boxes or exiting an application.

For example, to get the application use the following:

```
var object app
app = this.application@
```

See [BaseClass Methods](#) for more information.

<- application_name@

Returns application object name

Class BaseClass get

Format appName = this.application_name@

Description The get method application_name@ returns the application object name as a string.

For example, to get the application object name use the following:

```
var appName
appName = this.application_name@
```

See [BaseClass Methods](#) for more information.

<- child@

Returns a child object for a passed text string

Class BaseClass get

Format object = this.child@(name)

Arguments name A string for the child object name.

Description The get method `child@` returns a child object with the same name as a passed string. The method throws an error if the child object does not exist. Use the get method `is_child@` to verify the child object exists.

For example, to get a child named Button use the following:

```
var object childObject
childObject = this.child@("Button")
```

See [BaseClass Methods](#) for more information.

<- children@

Returns an array of child objects

Class BaseClass get

Format objectArray = this.children@

Description The get method `children@` returns an array of immediate child objects, an empty array if the object has no children.

For example, to get the child objects of the current object use the following:

```
var object childObjects
childObjects = this.children@
```

See [BaseClass Methods](#) for more information.

<- class_library@

Returns a class object reference

Class BaseClass get

Format classObjectRef = this.class_library@(libName)

Arguments libName The name of the .dll file that you loaded into Builder that defines the class.

Description The `class_library@` method is used when you create an object from a user-defined class. The user-defined class is loaded into Builder from a .dll file.

The classObjectRef value returned by this method is an object reference to a class object. You should pass this class object reference to the macro OBJECT_CREATE@ to instantiate a new builder object.

See also [OBJECT_CREATE@](#)

<- data_set_find@

Returns an object reference to a named dataset object

Class BaseClass get

Format object var = this.data_set_find@(objectName)

Arguments objectName The name of a dataset object

Description Returns an object reference to the named DatasetClass object. This object must be part of the Builder application, otherwise the method returns a NULL value.

<- comments@

Returns application comments

Class BaseClass get

Format stringArray = this.comments@

Description The get method comments@ returns the comments stored with an application as an array of strings.

For example, to get the application comments use the following:

```
var stringArray
stringArray = this.comments@
```

See [BaseClass Methods](#) for more information.

<- inherit@

Returns the inheritance class

Class BaseClass get

Format object = this.inherit@

Description The get method inherit@ returns the inheritance class. The inheritance class defines the methods and events available for the current object. The current object is an instance of the inheritance class.

For example, to get the inheritance class of the current object use the following:

```
var object classObject
classObject = this.inherit@
```

See [BaseClass Methods](#) for more information.

<- is_child@

Returns child status

Class BaseClass get

Format flag = this.is_child@(name)

Arguments name A string for the child object name.

Description The get method is_child@ returns child status as a Boolean value. The method returns TRUE if the object is a child of the current object. The method returns FALSE if the object is not a child of the current object or does not exist.

For example, to get a child status use the following:

```
var flag
flag = this.is_child@("Button")
```

See [BaseClass Methods](#) for more information.

<- load_methods_file@

Sets the methods to a source file

Class BaseClass set

Format retVal = obj.load_methods_file@(fileName[,dumpError])

Arguments fileName A string for the full pathname of the source file, or an array of strings containing the method source.

`dumpError` An optional Boolean argument, TRUE means dump the compilation errors into the Array List dialog box and display the dialog box. FALSE means return compilation errors in a two-dimensional array. The default value is FALSE.

`retVal[0]` contains the object's hierarchy, beginning with the object, then its parent and all other objects that it descends from. `retVal[1]` contains the array of error strings.

Description The get method `load_methods_file@` sets the available methods for an object to the methods contained in a passed source file and returns any compilation errors. If no compilation errors occur, or if you dump the errors, the methods returns NULL. The source file contains ASCII text defining methods. Use this method if want want to handle compilation errors within your application code.

For example, to set the current object methods to the methods defined in the source file `/user/Applixware/build_method` use the following:

```
var object button
var retVal
button = this.child@("Button")
retVal = button.load_methods_file@("/user/Applixware/build_method")
IF retVal <> NULL
{...
    code to handle a compilation error
...}
```

To set the current object methods with an array of strings use the following:

```
var object button
button = this.child@("Button")
retVal = this.load_methods_file@ = {
    "@@@OBJECTS",
    "set initialize_event",
    "endset"
}
```

See also set [load_methods_file@](#)

See [BaseClass Methods](#) for more information.

<- name@

Returns the name of the object as a text string

Class BaseClass get

Format string = this.name@

Description The get method name@ returns the name of the object.

For example, to retrieve the name of the current object as a string variable use the following:

```
var string
string = this.name@
```

If the object's name is Button2, the string variable contains the text string Button2.

See [BaseClass Methods](#) for more information.

<- parent@

Returns the parent object

Class BaseClass get

Format object = this.parent@

Description The get method parent@ returns the parent object. Use the parent object to perform operations not accessible with the current object.

For example, to get the parent object of the current object use the following:

```
var object parentObject
parentObject = this.parent@
```

See [BaseClass Methods](#) for more information.

<- public_objvars@

Returns public objvars

Class BaseClass get

Format vars = this.public_objvars@

Description The get method public_objvars@ returns the public objvars of an object. Public objvars are object variables you can reference from other objects without using methods. Use the set method public_objvars@ to define the variables you can set or get without methods.

For example, to get the public objvars of an object use the following:

```
var variables
variables = this.public_objvars@
```

See [BaseClass Methods](#) for more information.

<- quick_help@

Returns quick help string for a method

Class BaseClass get

Format helpStr = this.quick_help@(type, name)

Arguments type The type of method: set or get.
name A method name in the class.

Description The get method quick_help@ returns the quick help string for a method in the class. Use this method with user-defined classes to determine if a quick help string is assigned to the method. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name.

See also set [quick_help@](#)
set [quick_help_initialize_event](#)

See [BaseClass Methods](#) for more information.

<- resource_content@

Returns contents of a resource file

Class BaseClass get

Format info = this.resource_content@(file)

Arguments file The resource file path name.

Description The get method `resource_content@` returns the contents of a resource file. Resource files are loaded into an application with the **Load Resources** dialog box. The method returns NULL if the resource file does not exist or if it is not loaded as a resource.

Resource information is saved as part of the top-level ApplicationClass object. Use this method with the ApplicationClass object to get the contents of a resource file.

See **BaseClass Methods** for more information.

<- sibling@

Returns an object sharing the same parent

Class BaseClass get

Format object = this.sibling@(name)

Arguments name A string for the sibling object name.

Description The get method `sibling@` returns an object sharing the same parent as the current object. The method throws an error if the sibling does not exist.

For example, to get a sibling named `brotherButton` use the following:

```
var object siblingObject
siblingObject = this.sibling@("brotherButton")
```

See **BaseClass Methods** for more information.

<- source_strings@

Returns an object's source

Class BaseClass get

Format stringArray = this.source_strings@

Description The get method `source_strings@` returns an object's source as an array of strings. Use the method to determine the methods and events defined for the object.

For example, to get an object's source use the following:

```
var stringArray
```

```
stringArray = this.source_strings@
```

See [BaseClass Methods](#) for more information.

<- timer@

Returns an object's timer setting

Class BaseClass get

Format time = this.timer@

Description The get method timer@ returns an object's timer setting, in seconds. The timer value determines the frequency of when a [time out event](#) is generated. Use the set method [timer@](#) to set the time-out interval.

See [BaseClass Methods](#) for more information.

-> add_child@

Adds an object as a child

Class BaseClass set

Format this.add_child@(object)

Arguments object An object created with the [OBJECT CREATE@](#) macro.

Description The set method add_child@ adds an object to the current object's list of children. For example, to add an object buttonExample to the current object use the following:

```
this.add_child@(buttonExample)
```

See [BaseClass Methods](#) for more information.

>- application_name@

Sets application object name

Class BaseClass set

Format this.application_name@(name)

Arguments name A string for the application object name.

Description The set method `application_name@` sets the application object name to the passed string.

For example, to set the application object name use the following:

```
var appName
this.application_name@("Application")
```

See [BaseClass Methods](#) for more information.

-> **comments@**

Sets application comments

Class BaseClass set

Format `this.comments@(stringArray)`

Arguments stringArray A single string, or array of strings

Description The set method `comments@` sets the comments stored with an application to the passed string or array of strings.

For example, to set the application comments use the following:

```
var stringArray
stringArray[0] = "First line of comments"
stringArray[1] = "Last line of comments"
this.comments@(stringArray)
```

See [BaseClass Methods](#) for more information.

-> **delete_child@**

Removes a child object

Class BaseClass set

Format `this.delete_child@(objectName)`

Arguments objectName The child object to delete.

Description The set method `delete_child@` removes the passed child object from the current object. Use the get method `child@` to get a pointer to the object, then pass the returned object to `delete_child@`.

For example, to remove the child object `exampleButton` use the following:

```
var object childObject
childObject = this.child@("exampleButton")
this.delete_child@(childObject)
```

See [BaseClass Methods](#) for more information.

-> `delete_children@`

Removes all child objects

Class BaseClass set

Format `this.delete_children@`

Description The set method `delete_children@` removes all child objects from the current object.

For example, to remove all child objects from the current object use the following:

```
this.delete_children@
```

See [BaseClass Methods](#) for more information.

-> `inherit@`

Changes the inherit class

Class BaseClass set

Format `this.inherit@(className)`

Arguments `className` A string for the class name.

Description The set method `inherit@` changes the inherit class of the current object to the passed class name.

For example, to change the inherit class of the current object to `ButtonClass` use the following:

```
this.inherit@("ButtonClass")
```

or

```
this.inherit@ = "ButtonClass"
```

See [BaseClass Methods](#) for more information.

-> load_methods_file@

Sets the methods to a source file

Class BaseClass set

Format obj.load_methods_file@(fileName)

Arguments fileName A string for the full pathname of the source file, or an array of strings.

Description The set method load_methods_file@ sets the available methods for an object to the methods contained in a passed source file. The source file contains ASCII text defining methods.

For example, to set the current object methods to the methods defined in the source file /user/Applixware/build_method use the following:

```
var object button
button = this.child@("Dismiss")
button.load_methods_file@("/user/Applixware/build_method")
```

To set the current object methods with an array of strings use the following:

```
var object button
button = this.child@("Dismiss")
button.load_methods_file@ = {
    "@@@"OBJECTS",
    "set initialize_event",
    "endset"
}
```

See also get [load_methods_file@](#)

See [BaseClass Methods](#) for more information.

-> name@

Sets the name of the object

Class BaseClass set

Format this.name@(string)

Arguments string A string for the object name.

Description The set method name@ sets the name of the object to the passed string.

For example, to set the name of the current object to CancelButton use the following:

```
    this.name@ = "CancelButton"
```

See [BaseClass Methods](#) for more information.

-> parent@

Sets the parent object

Class BaseClass get

Format this.parent@(object)

Arguments object An object in the application.

Description The set method parent@ sets the parent object of the current object.

For example, to set the parent object of the current object use the following:

```
    var object parentObject
    parentObject = this.application@.child@("Form")
    this.parent@(parentObject)
```

See [BaseClass Methods](#) for more information.

-> public_objvars@

Sets public objvars

Class BaseClass set

Format `this.public_objvars@(nameArray)`

Arguments `nameArray` An array of object names.

Description The set method `public_objvars@` sets the public objvars. Public objvars are object variables you can reference from other objects without using methods. Use the method to define the variables you can set or get without methods.

For example, to define object variables as public objvars, use the following:

```
objvar objvar1
```

```
objvar objvar2
```

```
this.public_objvars@ = {"objvar1", "objvar2"}
```

See [BaseClass Methods](#) for more information.

-> quick_help@

Sets quick help string for a method

Class BaseClass set

Format `this.quick_help@(type, name, helpStr)`

Arguments `type` The type of method: set or get.

`name` A method name in the class.

`helpStr` The quick help string

Description The set method `quick_help@` sets the quick help string for a method in the class. Use this method in the [quick_help_initialize_event](#) of user-defined classes to assign a quick help string to a method. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name.

See also get [quick_help@](#)

See [BaseClass Methods](#) for more information.

-> timer@

Sets object time-out timer

Class BaseClass set

Format this.timer@(time)

Arguments time A time value, in seconds.

Description The set method timer@ sets the object's time-out timer. Use the timer to generate a [time_out_event](#). A time_out_event is generated when the object's timer expires.

For example, a clock application would use this method to set the timer to 1 second. A time_out_event would occur after 1 second, then the code in the time_out_event would set the new time and display it.

See [BaseClass Methods](#) for more information.

<- error_event

Called for system errors

Class BaseClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [BaseClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class BaseClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [BaseClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class BaseClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [BaseClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class BaseClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [BaseClass Methods](#) for more information.

-> quick_help_initialize_event

Initializes quick help strings for class methods

Class BaseClass event

Format this.quick_help_initialize_event

Description The quick_help_initialize_event initializes the quick help strings for methods in a user-defined class. The quick help string appears in the Quick Help area in the Class Browser when you choose the method name. Use the set method [quick_help@](#) in this event to define the quick help strings for methods in the class.

See also get [quick_help@](#)

See [BaseClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class CanvasClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the canvas. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [CanvasClass Methods](#) for more information.

<- height@

Returns canvas height

Class CanvasClass get

Format pixels = this.height@

Description The get method height@ returns the canvas height in pixels. You can use this method with the set method height@ to verify and change the object's height.

For example, to make the canvas height at least 100 pixels you would use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 100
    this.height@ = 100         'set method
```

See [CanvasClass Methods](#) for more information.

<- hscroll_is_enabled@

Returns horizontal scroll bar status

Class CanvasClass get

Format flag = this.hscroll_is_enabled@

Description The get method `hscroll_is_enabled@` returns the horizontal scroll bar status as a Boolean value. The method returns TRUE if the horizontal scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

See [CanvasClass Methods](#) for more information.

<- hscroll_length@

Returns horizontal scroll bar length

Class CanvasClass get

Format pixels = this.hscroll_length@

Description The get method `hscroll_length@` returns the horizontal scroll bar length, in pixels.

See [CanvasClass Methods](#) for more information.

<- hscroll_origin@

Returns horizontal scroll bar origin

Class CanvasClass get

Format pixels = this.hscroll_origin@

Description The get method `hscroll_origin@` returns the horizontal scroll bar origin, in pixels, from the bottom left corner of the canvas.

See [CanvasClass Methods](#) for more information.

<- icon_size@

Returns icon size

Class CanvasClass get

Format sizeArray = this.icon_size@(object pen, name)

Arguments

pen	A pen object.
name	The icon name, less the file extension. The icon is a .im file located in your ELF search path, or a glom file loaded as a resource.

Description The get method icon_size@ returns the size of an icon in the canvas drawn with a given pen object. The size is returned as a two element array. sizeArray[0] is the icon width, in pixels, and sizeArray[1] is the icon height, in pixels. The size of the icon on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

<- inset_file@

Returns inset file name

Class CanvasClass get

Format fileName = this.inset_file@

Description The get method inset_file@ returns the file name of the inset file.

See [CanvasClass Methods](#) for more information.

<- inset_file@

Returns inset file name

Class CanvasClass get

Format fileName = this.inset_file@

Description The get method inset_file@ returns the file name of the inset file.

See [CanvasClass Methods](#) for more information.

<- inset_type@

Returns inset file type

Class CanvasClass get

Format applID = this.inset_type@

Description The get method `inset_type@` returns the application type of the canvas inset. The valid inset types are:

- 1 Applixware Words
- 2 Applixware Graphics
- 3 Applixware Spreadsheets

See [CanvasClass Methods](#) for more information.

<- measure_inset_object@

Returns size of inset

Class CanvasClass get

Format retVal = this.measure_inset_object@([format ax_units@ units[, format ax_area@ area]])

Arguments

units	The measurement units for the inset, in <code>ax_units@</code> format.
area	The inset area information, in <code>ax_area@</code> format.

Description The get method `measure_inset_object@` returns the size of the inset in a 2-element array. `retVal[0]` is the units information, in `ax_units@` format. `retVal[1]` is the area information, in `ax_area@` format. The formats are defined in the `install_dir/axdata/elf/insets_.am` header file, include this file in your object method source.

See [CanvasClass Methods](#) for more information.

<- scroll_pos@

Returns canvas position

Class CanvasClass get

Format posArray = this.scroll_pos@

Description The get method scroll_pos@ returns the scroll position of the canvas. The position is returned as a two element array. posArray[0] is the x-position of the canvas, in pixels, relative to the (0,0) location of the upper left corner of the canvas. posArray[1] is the y-position of the canvas, in pixels, relative to the (0,0) location of the upper left corner of the canvas. Use the method with the set method scroll_pos@ to verify and set the canvas position.

See [CanvasClass Methods](#) for more information.

<- text_font_baseline@

Returns text width

Class CanvasClass get

Format width = this.text_font_baseline@(object pen)

Arguments pen A pen object.

Description The get method text_font_baseline@ returns the baseline (the amount of space between lines of text) in the canvas drawn with a given pen object. The baseline is returned as a value in pixels. The baseline of the text on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

<- text_font_height@

Returns text width

Class CanvasClass get

Format height = this.text_font_height@(object pen)

Arguments pen A pen object.

Description The get method text_font_height@ returns the height of text in the canvas drawn with a given pen object. The height is returned as a value in pixels. The height of the text on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

<- text_width@

Returns text width

Class CanvasClass get

Format width = this.text_width@(object pen, string)

Arguments pen A pen object.
string A text string.

Description The get method text_width@ returns the width of a text string in the canvas drawn with a given pen object. The width is returned as a value in pixels. The width of the text on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

<- thickness@

Returns canvas thickness

Class CanvasClass get

Format pixels = this.thickness@

Description The get method height@ returns the canvas thickness in pixels. The thickness is the bevel appearance of the canvas sides.

See [CanvasClass Methods](#) for more information.

<- vscroll_is_enabled@

Returns vertical scroll bar status

Class CanvasClass get

Format flag = this.vscroll_is_enabled@

Description The get method `vscroll_is_enabled@` returns the vertical scroll bar status as a Boolean value. The method returns TRUE if the vertical scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

See [CanvasClass Methods](#) for more information.

<- vscroll_length@

Returns vertical scroll bar length

Class CanvasClass get

Format pixels = this.vscroll_length@

Description The get method `vscroll_length@` returns the vertical scroll bar length, in pixels.

See [CanvasClass Methods](#) for more information.

<- vscroll_origin@

Returns vertical scroll bar length

Class CanvasClass get

Format pixels = this.vscroll_origin@

Description The get method `vscroll_origin@` returns the vertical scroll bar origin, in pixels, from the top right corner of the canvas.

See [CanvasClass Methods](#) for more information.

<- width@

Returns canvas width

Class CanvasClass get

Format pixels = this.width@

Description The get method width@ returns the canvas width in pixels. You can use this method with the set method width@ to verify and change the object's width.

For example, to make the canvas width at least 250 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [CanvasClass Methods](#) for more information.

-> button3_menu_info@

Sets pop up menu info

Class CanvasClass set

Format this.button3_menu_info@(format arrayof rminfo@ info)

Arguments info An array of rminfo@ information. The rminfo@ format is defined in the dialog_.am file, located in the *install_dir/axdata/elf* directory. The rminfo@ format is defined as:

format rminfo@

name, The name displayed in the menu.

macro_name, The macro or method name. Macro names must begin with the @ character to indicate it is a macro, not a method.

args, An argument string.

active A Boolean value, TRUE means the menu option is enabled, FALSE means the menu option is grayed and disabled.

Description The set method `button3_menu_info@` sets the pop up menu information. A pop up menu appears with a right mouse button press. A pop up menu is a free-floating menu associated with a dialog box control. A pop up menu can be activated when the mouse pointer is in the control area.

See [CanvasClass Methods](#) for more information.

-> `clear@`

Clears a canvas area

Class CanvasClass set

Format `this.clear@(object pen, start, area)`

Arguments

pen	A pen object.
start	A two element array for the starting point. <code>start[0]</code> is the start point x position, in pixels. <code>start[1]</code> is the start point y position, in pixels.
area	A two element array for the area dimensions. <code>area[0]</code> is the width of the area, in pixels. <code>area[1]</code> is the height of the area, in pixels.

Description The set method `clear@` clears a canvas area for the passed pen, start position, and area arguments.

See [CanvasClass Methods](#) for more information.

-> `clear_area@`

Clears a canvas area

Class CanvasClass set

Format `this.clear_area@(object pen, x, y, width, height)`

Arguments

pen	A pen object.
x	The starting x position, in pixels.
y	The starting y position, in pixels.

width	The width of the area, in pixels. If the width is 0, the width is set to this.width -x.
height	The height of the area, in pixels. If the height is 0, the width is set to this.height -y.

Description The set method `clear_area@` clears a canvas area for the passed pen, start, and area arguments.

See [CanvasClass Methods](#) for more information.

-> control_color@

Sets control color

Class CanvasClass set

Format `this.control_color@(type, c1, c2, c3, c4)`

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.

c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values.

See [CanvasClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class CanvasClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values.

See [CanvasClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class CanvasClass set

Format `this.control_color_is_workspace@(flag)`

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method `control_color_is_workspace@` sets the color used by the object.

See [CanvasClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class CanvasClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name.

See [CanvasClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class CanvasClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the control color with RGB values.

See [CanvasClass Methods](#) for more information.

-> create_inset@

Creates the inset object

Class CanvasClass set

Format this.create_inset@(format ax_inset_info@ inset, app_inset)

Arguments inset The inset information, in ax_inset_object@ format. The format is defined in the insets_.am file, located in the *install_dir/axdata/elf* directory.

app_inset The inset information, defined in the *_inset_source@ format for the source file. The wp_inset_source@ format is defined in the wp_.am header file. The gr_inset_source@ format is defined in the graphic_.am header file. The ss_inset_source@ format is defined in the spsheet_.am header file.

Description The set method create_inset@ creates the inset object, but does not render the inset on the canvas.

See [CanvasClass Methods](#) for more information.

-> draw_arc@

Draws an arc

Class CanvasClass set

Format this.draw_arc@(object pen, x, y, width, height, startAngle, endAngle)

Arguments

pen	A pen object.
x	The upper left corner x position, in pixels, of the rectangle containing the arc.
y	The upper left corner y position, in pixels, of the rectangle containing the arc.
width	The width, in pixels, of the rectangle containing the arc. The width is also known as the major axis of the arc.
height	The height, in pixels, of the rectangle containing the arc. The height is also known as the minor axis of the arc.
startAngle	The starting angle, in quad degrees, of the arc, relative to the 3'o clock position from center.
endAngle	The ending angle, in quad degrees, of the arc, relative to the 3'o clock position from center. 1440 quad degrees is a complete circle.

Description The set method draw_arc@ draws an arc on the canvas. The appearance of the arc on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> draw_icon@

Draws an icon

Class CanvasClass set

Format this.draw_icon@(object pen, point, NULL, name)

Arguments

pen	A pen object.
point	A two element array for the upper left corner of the icon. point[0] is the x position, in pixels. point[1] is the y position, in pixels.
name	The icon name, less the file extension. The icon is a .im Applixware bit-map file or an image in a glom file. The files are located in your ELF search path, or are loaded as a resource in the Builder file.

Description The set method draw_icon@ draws an icon on the canvas.

See [CanvasClass Methods](#) for more information.

-> draw_inset@

Creates and draws the inset object

Class CanvasClass set

Format this.draw_inset@(format ax_inset_info@ inset, app_inset)

Arguments

inset	The inset information, in ax_inset_object@ format. The format is defined in the insets_.am file, located in the <i>install_dir/axdata/elf</i> directory.
app_inset	The inset information, defined in the *_inset_source@ format for the source file. The wp_inset_source@ format is defined in the wp_.am header file. The gr_inset_source@ format is defined in the graphic_.am header file. The ss_inset_source@ format is defined in the spsheet_.am header file.

Description The set method draw_inset@ creates the inset object and draws it on the canvas.

See [CanvasClass Methods](#) for more information.

-> draw_line@

Draws a line

Class CanvasClass set

Format this.draw_line@(object pen, startPoint, endPoint)

Arguments

pen	A pen object.
startPoint	A two element array for the start point of the line. point[0] is the x position, in pixels. point[1] is the y position, in pixels.
endPoint	A two element array for the end point of the line. point[0] is the x position, in pixels. point[1] is the y position, in pixels.

Description The set method draw_line@ draws a line on the canvas. The appearance of the line on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> draw_line_from_xys@

Draws a line

Class CanvasClass set

Format this.draw_line_from_xys@(object pen, xStart, yStart, xEnd, yEnd)

Arguments

pen	A pen object.
xStart	The starting x position, in pixels.
yStart	The starting y position, in pixels.
xEnd	The ending x position, in pixels.
yEnd	The ending y position, in pixels.

Description The set method draw_line_from_xys@ draws a line on the canvas. The appearance of the line on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> draw_polygon@

Draws a polygon

Class CanvasClass set

Format this.draw_polygon@(object pen, points, quantity)

Arguments

pen	A pen object.
points	An array of points. Each point is an array of two elements. points[z,0] is the x position, in pixels. points[z,1] is the y position, in pixels.
quantity	The number of points in the points array.

Description The set method draw_polygon@ draws a polygon on the canvas. The appearance of the polygon on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> draw_rectangle@

Draws a rectangle

Class CanvasClass set

Format this.draw_rectangle@(object pen, x, y, width, height)

Arguments

pen	A pen object.
x	The upper left corner x position, in pixels, of the rectangle.
y	The upper left corner y position, in pixels, of the rectangle.
width	The width, in pixels, of the rectangle.
height	The height, in pixels, of the rectangle.

Description The set method draw_rectangle@ draws a rectangle on the canvas. The appearance of the rectangle on the canvas is dependent on the pen object attributes. See [Pen-Class Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> draw_text@

Draws text

Class CanvasClass set

Format this.draw_text@(object pen, point, NULL, text)

Arguments

pen	A pen object.
point	A two element array for the upper left corner of the text string. point[0] is the x position, in pixels. point[1] is the y position, in pixels.
text	A text string.

Description The set method draw_text@ draws text on the canvas. The appearance of the text on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> fill_arc@

Draws a filled arc

Class CanvasClass set

Format this.fill_arc@(object pen, x, y, width, height, startAngle, endAngle)

Arguments

pen	A pen object.
x	The upper left corner x position, in pixels, of the rectangle containing the arc.
y	The upper left corner y position, in pixels, of the rectangle containing the arc.
width	The width, in pixels, of the rectangle containing the arc. The width is also known as the major axis of the arc.
height	The height, in pixels, of the rectangle containing the arc. The height is also known as the minor axis of the arc.
startAngle	The starting angle, in quad degrees, of the arc, relative to the 3'o clock position from center.

endAngle The ending angle, in quad degrees, of the arc, relative to the 3'o clock position from center. 1440 quad degrees is a complete circle.

Description The set method fill_arc@ draws a filled arc on the canvas. The appearance of the filled arc on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> fill_polygon@

Draws a filled polygon

Class CanvasClass set

Format this.fill_polygon@(object pen, points, quantity)

Arguments

pen	A pen object.
points	An array of points. Each point is an array of two elements. points[z,0] is the x position, in pixels. points[z,1] is the y position, in pixels.
quantity	The number of points in the points array.

Description The set method fill_polygon@ draws a filled polygon on the canvas. The appearance of the filled polygon on the canvas is dependent on the pen object attributes. See [Pen-Class Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> fill_rectangle@

Draws a filled rectangle

Class CanvasClass set

Format this.fill_rectangle@(object pen, x, y, width, height)

Arguments

pen	A pen object.
x	The upper left corner x position, in pixels, of the rectangle.
y	The upper left corner y position, in pixels, of the rectangle.
width	The width, in pixels, of the rectangle.
height	The height, in pixels, of the rectangle.

Description The set method `fill_rectangle@` draws a filled rectangle on the canvas. The appearance of the filled rectangle on the canvas is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [CanvasClass Methods](#) for more information.

-> height@

Sets canvas height

Class CanvasClass set

Format `this.height@(value)`

Arguments value The height, in pixels, of the canvas.

Description The set method `height@` sets the canvass's height. You can use this method with the get method `height@` to verify and change the object's height.

For example, to make the canvas height at least 100 pixels you would use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 100
    this.height@ = 100         'set method
```

See [CanvasClass Methods](#) for more information.

-> hscroll_is_enabled@

Sets horizontal scroll bar status

Class CanvasClass set

Format `this.hscroll_is_enabled@(flag)`

Arguments flag A Boolean value. TRUE makes the horizontal scroll bar enabled and visible. FALSE makes the scroll bar disabled and hidden.

Description The set method `hscroll_is_enabled@` sets the horizontal scroll bar status.

See [CanvasClass Methods](#) for more information.

-> `hscroll_length@`

Sets horizontal scroll bar length

Class CanvasClass set

Format `this.hscroll_length@(pixels)`

Arguments pixels The length of the scroll bar, in pixels

Description The set method `hscroll_length@` sets the horizontal scroll bar length.

See [CanvasClass Methods](#) for more information.

-> `hscroll_origin@`

Sets horizontal scroll bar origin

Class CanvasClass set

Format `this.hscroll_origin@(pixels)`

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the canvas.

Description The set method `hscroll_origin@` sets the horizontal scroll bar origin.

See [CanvasClass Methods](#) for more information.

-> `inset_file@`

Sets the inset file

Class CanvasClass set

Format `this.inset_file@(fileName)`

Arguments fileName The file name of the inset

Description The set method `inset_file@` sets the inset file. The method also sets the inset type, based upon the file type of the passed file name.

See [CanvasClass Methods](#) for more information.

-> paint_height@

Sets drawing area height

Class CanvasClass set

Format this.paint_height@(height)

Arguments height The canvas width, in pixels.

Description The set method paint_height@ sets the canvas drawing area height.

See [CanvasClass Methods](#) for more information.

-> paint_width@

Sets drawing area width

Class CanvasClass set

Format this.paint_width@(width)

Arguments width The canvas width, in pixels.

Description The set method paint_width@ sets the canvas drawing area width.

See [CanvasClass Methods](#) for more information.

-> scroll_incr@

Sets scroll bar increment

Class CanvasClass set

Format this.scroll_incr@(x,y)

Arguments x The x-axis scroll increment, in pixels.

y The y-axis scroll increment, in pixels.

Description The set method `scroll_incr@` sets the canvas scroll bar increment. The increment is the distance the canvas scrolls with a click on a scroll bar arrow. The default scroll bar increment is 25 pixels

See [CanvasClass Methods](#) for more information.

-> `scroll_page_incr@`

Sets page scroll increment

Class CanvasClass set

Format `this.scroll_page_incr@(x,y)`

Arguments `x` The x-axis scroll increment, in pixels.
`y` The y-axis scroll increment, in pixels.

Description The set method `scroll_page_incr@` sets the canvas page scroll increment. The increment is the distance the canvas scrolls with a click within a scroll bar. The default page scroll increment is the dimension of the canvas.

See [CanvasClass Methods](#) for more information.

-> `scroll_pos@`

Sets scroll bar position

Class CanvasClass set

Format `this.scroll_pos@(point)`

Arguments `point` A two element array for the position. `point[0]` is the x position, in pixels, from the upper left corner of the canvas. `point[1]` is the y position, in pixel, from the upper left corner of the canvas.

Description The set method `scroll_pos@` sets the canvas scroll bar position.

See [CanvasClass Methods](#) for more information.

-> **thickness@**

Sets canvas thickness

Class CanvasClass set

Format this.thickness@(pixels)

Arguments pixels The thickness, in pixels.

Description The set method `height@` sets the canvas thickness in pixels. The thickness is the bevel appearance of the canvas sides.

See [CanvasClass Methods](#) for more information.

-> **vscroll_is_enabled@**

Sets vertical scroll bar status

Class CanvasClass set

Format this.vscroll_is_enabled@(flag)

Arguments flag A Boolean value. TRUE makes the vertical scroll bar enabled and visible. FALSE makes the scroll bar disabled and hidden.

Description The set method `vscroll_is_enabled@` sets the vertical scroll bar status.

See [CanvasClass Methods](#) for more information.

-> **vscroll_length@**

Sets vertical scroll bar length

Class CanvasClass set

Format this.vscroll_length@(pixels)

Arguments pixels The length of the scroll bar, in pixels

Description The set method `vscroll_length@` sets the vertical scroll bar length.

See [CanvasClass Methods](#) for more information.

-> **vscroll_origin@**

Sets vertical scroll bar origin

Class CanvasClass set

Format this.vscroll_origin@(pixels)

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the canvas.

Description The set method vscroll_origin@ sets the vertical scroll bar origin.

See [CanvasClass Methods](#) for more information.

-> **width@**

Sets canvas width

Class CanvasClass set

Format this.width@(value)

Arguments value The width, in pixels, of the canvas.

Description The set method width@ sets the canvass's width. You can use this method with the get method width@ to verify and change the object's width.

For example, to make the canvas width at least 250 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [CanvasClass Methods](#) for more information.

-> **button2_double_click_event**

Called when middle mouse button double-clicked

Class CanvasClass event

Format this.button2_double_click_event(point, shifted, controlled)

Arguments

point	A two element array for the position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The button2_double_click_event is called by Applixware Builder when the middle mouse button is double-clicked. The button2_press_event event is called first, then the button2_double_click_event. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> **button2_motion_event**

Called when mouse drag action with middle mouse button occurs

Class CanvasClass event

Format this.button2_motion_event(point, shifted, controlled)

Arguments

point	A two element array for the most recent position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The button2_motion_event is called by Applixware Builder when a mouse drag action with the middle mouse button occurs. This event is called after a button2_press_event,

after the mouse stops moving or the mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> **button2_press_event**

Called when middle mouse button pressed

Class CanvasClass event

Format this.button2_press_event(point, shifted, controlled)

Arguments

point	A two element array for the position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed. While controlled is TRUE, the method expects more point values to be passed.

Description The button2_press_event is called by Applixware Builder when the middle mouse button is pressed. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> **button2_release_event**

Called when middle mouse button released

Class CanvasClass event

Format this.button2_release_event(point, shifted, controlled)

Arguments

point	A two element array for the position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.

controlled A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The `button2_release_event` is called by Applixware Builder when the middle mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> `button3_double_click_event`

Called when right mouse button double-clicked

Class CanvasClass event

Format `this.button3_double_click_event(point, shifted, controlled)`

Arguments

<code>point</code>	A two element array for the position. <code>point[0]</code> is the x position, in pixels, from the upper left corner of the canvas. <code>point[1]</code> is the y position, in pixel, from the upper left corner of the canvas.
<code>shifted</code>	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
<code>controlled</code>	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The `button3_double_click_event` is called by Applixware Builder when the middle mouse button is double-clicked. The `button3_press_event` event is called first, then the `button3_double_click_event`. This is a user-defined event, place all actions you want performed in the event definition.

NOTE: This event is not called if popup menu information is set with the [button3_menu_info@](#) method.

See [CanvasClass Methods](#) for more information.

<- `button3_menu_state_event`

Sets menu state before pop up menu is displayed

Class CanvasClass event

Format `flag = this.button3_menu_state_event(function)`

Arguments function The name of a pop up menu function.

Description The `button3_menu_state_event` is called by Applixware Builder before a pop up menu is displayed for a control. A pop up menu is displayed when the right mouse button is pressed. The event should be programmed to return either a Boolean value or NULL. The event should return TRUE if the menu item for the function should be active. The event should return FALSE if the menu item for the function should be grayed and inactive. The event should return NULL if the state of the menu item is unchanged.

For example, the following event sets the state of different menu items:

```
get button3_menu_state_event(function)
    case of function
    case "menu_1", "menu_2"
        return(NULL) ' No change in status
    case "menu_3"
        return(TRUE) ' Active menu item
    case "menu_4"
        return(FALSE) ' Inactive menu item
    endcase
endget
```

Use the [button3 menu info@](#) method to set the pop up menu information.

NOTE: The Debugger cannot access break points set in the `button3_menu_state_event`. Do not use break points in the `button3_menu_state_event` while you are debugging an application.

See [CanvasClass Methods](#) for more information.

-> **button3_motion_event**

Called when mouse drag action with right mouse button occurs

Class CanvasClass event

Format `this.button3_motion_event(point, shifted, controlled)`

Arguments point A two element array for the most recent position. `point[0]` is the x position, in pixels, from the upper left corner of the canvas. `point[1]` is the y position, in pixel, from the upper left corner of the canvas.

- shifted A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
- controlled A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The `button3_motion_event` is called by Applixware Builder when a mouse drag action with right mouse button occurs. This event is called after a `button3_press_event`, after the mouse stops moving or the mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

NOTE: This event is not called if popup menu information is set with the [button3 menu info@](#) method.

See [CanvasClass Methods](#) for more information.

-> `button3_press_event`

Called when right mouse button pressed

Class CanvasClass event

Format `this.button3_press_event(point, shifted, controlled)`

- Arguments**
- `point` A two element array for the position. `point[0]` is the x position, in pixels, from the upper left corner of the canvas. `point[1]` is the y position, in pixel, from the upper left corner of the canvas.
 - `shifted` A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
 - `controlled` A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed. While `controlled` is TRUE, the method expects more point values to be passed.

Description The `button3_press_event` is called by Applixware Builder when the right mouse button is pressed. This is a user-defined event, place all actions you want performed in the event definition.

NOTE: This event is not called if popup menu information is set with the [button3 menu info@](#) method.

See [CanvasClass Methods](#) for more information.

-> **button3_release_event**

Called when right mouse button released

Class CanvasClass event

Format this.button3_release_event(point, shifted, controlled)

Arguments

point	A two element array for the position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The button3_release_event is called by Applixware Builder when the middle mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

NOTE: This event is not called if popup menu information is set with the [button3 menu info@](#) method.

See [CanvasClass Methods](#) for more information.

-> **button_press_event**

Called when left mouse button pressed

Class CanvasClass event

Format this.button_press_event(point, shifted, controlled)

Arguments

point	A two element array for the position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed. While controlled is TRUE, the method expects more point values to be passed.

Description The `button_press_event` is called by Appixware Builder when the left mouse button is pressed. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> `button_release_event`

Called when left mouse button released

Class CanvasClass event

Format `this.button_release_event(point, shifted, controlled)`

Arguments

<code>point</code>	A two element array for the position. <code>point[0]</code> is the x position, in pixels, from the upper left corner of the canvas. <code>point[1]</code> is the y position, in pixel, from the upper left corner of the canvas.
<code>shifted</code>	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
<code>controlled</code>	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The `button_release_event` is called by Appixware Builder when the left mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> `double_click_event`

Called when left mouse button double-clicked

Class CanvasClass event

Format `this.double_click_event(point, shifted, controlled)`

Arguments

<code>point</code>	A two element array for the position. <code>point[0]</code> is the x position, in pixels, from the upper left corner of the canvas. <code>point[1]</code> is the y position, in pixel, from the upper left corner of the canvas.
<code>shifted</code>	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.

controlled A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The `double_click_event` is called by Applixware Builder when the left mouse button is double-clicked. The `button_press_event` event is called first, then the `double_click_event`. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

<- error_event

Called for system errors

Class CanvasClass event

Format `flag = this.error_event(error_object)`

Arguments `error_object` An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [CanvasClass Methods](#) for more information.

-> expose_event

Called when new canvas area is exposed

Class CanvasClass event

Format `this.expose_event(point1, point2)`

Arguments `point1` A two element array for the upper left corner of the exposed area. `point[0]` is the x position, in pixels, from the upper left corner of the canvas. `point[1]` is the y position, in pixel, from the upper left corner of the canvas.

point2 A two element array for the lower right corner of the exposed area.
point[0] is the x position, in pixels, from the upper left corner of the canvas.
point[1] is the y position, in pixel, from the upper left corner of the canvas.

Description The expose_event is called by Applixware Builder when new canvas area is exposed, such as during a scroll. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> initialize_event

Called before displaying a dialog box control

Class CanvasClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the canvas dimensions, color, and so on.

See [CanvasClass Methods](#) for more information.

-> keyboard_event

Called when keyboard action occurs

Class CanvasClass event

Format this.keyboard_event(key, shifted, controlled)

Arguments

key	The ASCII key code.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The keyboard_event is called by Applixware Builder when a keyboard action occurs. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> motion_event

Called when mouse drag action occurs

Class CanvasClass event

Format this.motion_event(point, shifted, controlled)

Arguments

point	A two element array for the most recent position. point[0] is the x position, in pixels, from the upper left corner of the canvas. point[1] is the y position, in pixel, from the upper left corner of the canvas.
shifted	A Boolean value, TRUE means the SHIFT key is pressed, FALSE means the SHIFT key is not pressed.
controlled	A Boolean value, TRUE means the CONTROL key is pressed, FALSE means the CONTROL key is not pressed.

Description The motion_event is called by Applixware Builder when a mouse drag action occurs. This event is called after a button_press_event, after the mouse stops moving or the mouse button is released. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class CanvasClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [CanvasClass Methods](#) for more information.

-> scroll_event

Called when scroll occurs

Class CanvasClass event

Format this.scroll_event(point)

Arguments point A two element array for the position after the scroll. point[0] is the x position, in pixels, of the upper left corner of the canvas. point[1] is the y position, in pixel, of the upper left corner of the canvas.

Description The scroll_event is called by Applixware Builder when a scroll action occurs. The new position of the upper left corner is returned. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class CanvasClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [CanvasClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class CanvasClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is

set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The [timer@](#) method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [CanvasClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class CanvasClass event

Format this.update_event

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [CanvasClass Methods](#) for more information.

<- attributes@

Returns chart item attributes

Class ChartClass get

Format format gr_attribute@ info = this.attributes@(item)

Arguments item One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid
- minor vertical grid
- line, axis x<number>
- line, axis y<number>
- label, axis x<number>
- label, axis y<number>

```

tik label, axis x<number>
tik label, axis y<number>
data
label
data
title
subtitle
footer
legend box
legend title
legend labels

```

Description The get method `attributes@` returns the attributes for the given chart component in `gr_attribute@` format. The `gr_attribute@` format is defined in the `/install_dir/axdata/elf/graphic_.am` file. Include this file in any sources using the format.

The definition of `gr_attribute@` is as follows:

```
format gr_attribute@
```

```

format gr_fill_attr_type@ backfill,
format gr_fill_attr_type@ linefill,
format gr_shadow_attr_type@ shadow,
format gr_line_style_type@ line,
format gr_text_field_attr_type@ field,
format gr_text_char_attr_type@ char,

```

```
format gr_file_attr_type@
```

```

type,          'string; built-in <#>, <filename>, linear gradient
fg_color,     'string: name of color in colormap
bg_color,     'string: name of color in colormap
angle,        'int: in degrees
offset        'int: in mills (1000 mills = 1 inch)

```

```
format gr_shadow_attr_type@
```

```

type,          'string: none, background drop shadow, local drop shadow
color,         'string: name of color in colormap
               'int: in mills (1000 mills = 1 inch)
horizontal_offset
vertical_offset
               'int: in mills (1000 mills = 1 inch)

```

```
format gr_line_style_type@
```

```

style,
weight,
first_symbol,
join_symbol,
final_symbol

```

format gr_text_char_attr_type@
face,
size,
ruling,
strike_thru,
bold,
italic,
horizontal_scale,
vertical_offset,
horizontal_sub_sup,
vertical_sub_sup

format gr_text_field_attr_type@
horizontal_alignment,
vertical_alignment,
line_space,
horizontal_scale,
vertical_scale,
shear,
angle,
left_margin,
right_margin,
top_margin,
bottom_margin

In this format, the following elements are always NULL:

- field.line_space
- field.left_margin
- field.right_margin
- field.top_margin
- field.bottom_margin

See [ChartClass Methods](#) for more information.

<- attribute_back_fill@

Returns chart item background fill attributes

Class ChartClass get

Format attArray = this.attribute_back_fill@(item)

Arguments item One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid
- minor vertical grid
- line, axis x<number>
- line, axis y<number>
- label, axis x<number>
- label, axis y<number>
- tik label, axis x<number>
- tik label, axis y<number>
- data
- label
- data
- title
- subtitle
- footer
- legend box
- legend title
- legend labels

Description The get method `attribute_back_fill@` returns the background fill attributes for the given chart component as an array of 2 values. The array contains the following values:

`attArray[0]` The name of color in colormap for foreground color.
`attArray[1]` The name of color in colormap for background color.

See [ChartClass Methods](#) for more information.

<- attribute_character@

Returns chart item character attributes

Class ChartClass get

Format `attArray = this.attribute_character@(item)`

Arguments item One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid

minor vertical grid
line, axis x<number>
line, axis y<number>
label, axis x<number>
label, axis y<number>
tik label, axis x<number>
tik label, axis y<number>
data
label
data
title
subtitle
footer
legend box
legend title
legend labels

Description The get method `attribute_character@` returns the character attributes for the given chart component as an array of 4 items. The array contains the following values:

`attArray[0]` The font name.

`attArray[1]` The font point size.

`attArray[2]` The font bold state as a Boolean value, TRUE means the font is bold, otherwise it returns FALSE.

`attArray[3]` The font italic state as a Boolean value, TRUE means the font is italic, otherwise it returns FALSE.

See [ChartClass Methods](#) for more information.

<- attribute_field@

Returns chart item alignment

Class ChartClass get

Format `attArray = this.attribute_field@(item)`

Arguments item One of the following chart elements:

default
minor horizontal grid
major horizontal grid
major vertical grid
minor vertical grid

line, axis x<number>
line, axis y<number>
label, axis x<number>
label, axis y<number>
tik label, axis x<number>
tik label, axis y<number>
data
label
data
title
subtitle
footer
legend box
legend title
legend labels

Description The get method `attribute_field@` returns the alignment for the given chart component as an array of 2 items. The array contains the following values:

`attArray[0]` The horizontal alignment.

`attArray[1]` The vertical alignment.

The valid alignment values are:

0 Left

1 Center

2 Right

See [ChartClass Methods](#) for more information.

<- attribute_line_fill@

Returns chart item line fill attributes

Class ChartClass get

Format `attArray = this.attribute_line_fill@(item)`

Arguments `item` One of the following chart elements:

default
minor horizontal grid
major horizontal grid
major vertical grid
minor vertical grid

line, axis x<number>
line, axis y<number>
label, axis x<number>
label, axis y<number>
tik label, axis x<number>
tik label, axis y<number>
data
label
data
title
subtitle
footer
legend box
legend title
legend labels

Description The get method `attribute_line_fill@` returns the line fill attributes for the given chart component as an array of 2 values. The array contains the following values:

`attArray[0]` The name of color in colormap for foreground color.
`attArray[1]` The name of color in colormap for background color.

See [ChartClass Methods](#) for more information.

`<- attribute_line_style@`

Returns chart item line style attributes

Class ChartClass get

Format `attArray = this.attribute_line_style@(item)`

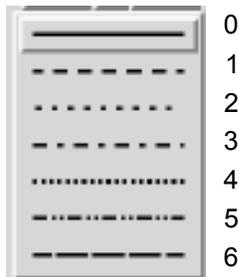
Arguments item One of the following chart elements:

default
minor horizontal grid
major horizontal grid
major vertical grid
minor vertical grid
line, axis x<number>
line, axis y<number>
label, axis x<number>
label, axis y<number>
tik label, axis x<number>
tik label, axis y<number>

data
 label
 data
 title
 subtitle
 footer
 legend box
 legend title
 legend labels

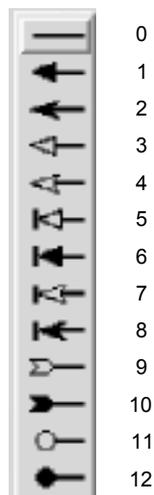
Description The get method `attribute_line_style@` returns the line style attributes for the given chart component as an array of 5 items. The array contains the following values:

`attArray[0]` The line style. Use one of the following numeric values:

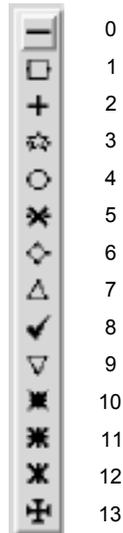


`attArray[1]` The line width, in pixels.

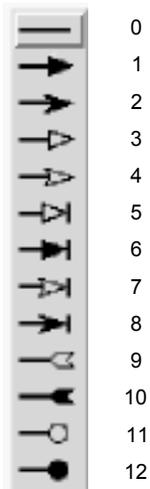
`attArray[2]` The symbol to be used at a line's beginning. Use one of the following numeric values:



`attArray[3]` The symbol to be used when lines are joined. Use one of the following numeric values:



attArray[4] The symbol used at the end of a line. Use one of the following numeric values:



See [ChartClass Methods](#) for more information.

<- attribute_shadow@

Returns chart item shadow attributes

Class ChartClass get

Format attArray = this.attribute_shadow@(item)

Arguments item One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid
- minor vertical grid
- line, axis x<number>
- line, axis y<number>
- label, axis x<number>
- label, axis y<number>
- tik label, axis x<number>
- tik label, axis y<number>
- data
- label
- data
- title
- subtitle
- footer
- legend box
- legend title
- legend labels

Description The get method `attribute_shadow@` returns the shadow attributes for the given chart component as an array of 2 items. The array contains the following values:

`attArray[0]` The shadow type. The valid types are:

- 0 none
- 1 background
- 2 local

`attArray[1]` The shadow color

See [ChartClass Methods](#) for more information.

<- axes_names@

Returns chart axes names

Class ChartClass get

Format axesNameArray = this.axes_names@

Description The get method `axes_names@` returns the names of the axes as an array of strings. The names of the axes take the following form:

```
axis x1
axis x2
...
axis y1
axis y2
...
```

See [ChartClass Methods](#) for more information.

`<- axis@`

Returns chart axis definition

Class ChartClass get

Format `format chart_axis_info@ info = this.axis@(name)`

Arguments name The name of one of the chart's axes. This name takes the following form:

```
axis x1
axis x2
...
axis y1
axis y2
...
```

Description The get method `axis@` returns the chart axis definition in `chart_axis_info@` format. The `chart_axis_info@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

`<- axis_auto_tik_flags@`

Returns axis auto tik settings

Class ChartClass get

Format `flagArray = this.axis_auto_tik_flags@(name)`

Arguments name The name of one of the chart's axes. This name takes the following form:

axis x1
axis x2
...
axis y1
axis y2
...

Description The get method `axis_auto_tik_flags@` returns the axis auto tik settings as an array of Boolean values. The array contains the following information:

`flagArray[0]` TRUE means automatically set major tik value.

`flagArray[1]` TRUE means automatically set minor tik value.

See [ChartClass Methods](#) for more information.

<- axis_auto_value_flags@

Returns axis auto value settings

Class ChartClass get

Format `flagArray = this.axis_auto_value_flags@(name)`

Arguments name The name of one of the chart's axes. This name takes the following form:

axis x1
axis x2
...
axis y1
axis y2
...

Description The get method `axis_auto_value_flags@` returns the axis auto value settings as an array of Boolean values. The array contains the following information:

`flagArray[0]` TRUE means automatically set maximum axis value.

`flagArray[1]` TRUE means automatically set minimum axis value.

`flagArray[2]` TRUE means automatically set base axis value.

See [ChartClass Methods](#) for more information.

<- axis_bar_spacing@

Returns axis bar spacing info

Class ChartClass get

Format valArray = this.axis_bar_spacing@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:

- axis x1
- axis x2
- ...
- axis y1
- axis y2
- ...

Description The get method `axis_bar_spacing@` returns the axis bar spacing information. The array contains the following information:

valArray[0] The bar margin.

valArray[1] The bar overlap.

See [ChartClass Methods](#) for more information.

<- axis_create@

Creates a chart axis

Class ChartClass get

Format format chart_axis_info@ info = this.axis_create@(type, format chart_axis_info@ info)

Arguments type One of the following strings:

- x Sets the X-axis as the primary axis.
- y Sets the Y-axis as the primary axis.

axisInfo A structure containing axis information in `chart_axis_info@` format. The `chart_axis_info@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format.

Description The get method `axis_create@` creates a chart axis and returns the axis name. The name is always of the form:

axis <x_or_y><number>

See [ChartClass Methods](#) for more information.

<- axis_destroy@

Deletes an axis from a chart

Class ChartClass get

Format remainingAxesArray = this.axis_destroy@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:

- axis x<number>
- axis y<number>

- axis x1
- axis x2
- ...
- axis y1
- axis y2
- ...

Description The get method axis_destroy@ deletes an axis from the chart and returns the remaining axis names as an array of strings. The names of the axes take the following form:

- axis x1
- axis x2
- ...
- axis y1
- axis y2
- ...

See [ChartClass Methods](#) for more information.

<- axis_label@

Returns chart axis label info

Class ChartClass get

Format valArray = this.axis_label@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:

axis x1
axis x2
...
axis y1
axis y2
...

Description The get method `axis_label@` returns the chart axis label information as an array of values. The array contains the following information:

`valArray[0]` The label string.
`valArray[1]` The label x-axis margin.
`valArray[2]` The label y-axis margin.
`valArray[3]` The label alignment:
0 Left
1 Center
2 Right
`valArray[4]` The spacing between labels and tik marks.

See [ChartClass Methods](#) for more information.

-> **attributes@**

Sets chart item attributes

Class ChartClass set

Format `this.attributes@(item, format gr_attribute@ info[, noDrawFlag])`

Arguments item One of the following chart elements:
default
minor horizontal grid
major horizontal grid
major vertical grid
minor vertical grid
line, axis x<number>
line, axis y<number>
label, axis x<number>
label, axis y<number>

tik label, axis x<number>
 tik label, axis y<number>
 data
 label
 data
 title
 subtitle
 footer
 legend box
 legend title
 legend labels

info The attributes being set for item.

noDrawFlag A Boolean value where TRUE indicates that the chart should not be re-drawn when the new attributes are applied.

Description The set method `attributes@` sets the attributes for the given chart component in `gr_attribute@` format. The `gr_attribute@` format is defined in the `/install_dir/axdata/elf/graphic_.am` file. Include this file in any sources using the format. If you are not setting (or using) an attribute within the format, set the attribute's value to NULL.

The definition of `gr_attribute@` is as follows:

format `gr_attribute@`

format `gr_fill_attr_type@` backfill,
 format `gr_fill_attr_type@` linefill,
 format `gr_shadow_attr_type@` shadow,
 format `gr_line_style_type@` line,
 format `gr_text_field_attr_type@` field,
 format `gr_text_char_attr_type@` char,

format `gr_file_attr_type@`

type, 'string; built-in <#>, <filename>, linear gradient
 fg_color, 'string: name of color in colormap
 bg_color, 'string: name of color in colormap
 angle, 'int: in degrees
 offset 'int: in mills (1000 mills = 1 inch)

format `gr_shadow_attr_type@`

type, 'string: none, background drop shadow, local drop shadow
 color, 'string: name of color in colormap
 'int: in mills (1000 mills = 1 inch)
 horizontal_offset
 vertical_offset
 'int: in mills (1000 mills = 1 inch)

format `gr_line_style_type@`

style,
weight,
first_symbol,
join_symbol,
final_symbol

format gr_text_char_attr_type@
face,
size,
ruling,
strike_thru,
bold,
italic,
horizontal_scale,
vertical_offset,
horizontal_sub_sup,
vertical_sub_sup

format gr_text_field_attr_type@
horizontal_alignment,
vertical_alignment,
line_space,
horizontal_scale,
vertical_scale,
shear,
angle,
left_margin,
right_margin,
top_margin,
bottom_margin

In this format, the following elements are always NULL:

- field.line_space
- field.left_margin
- field.right_margin
- field.top_margin
- field.bottom_margin

See [ChartClass Methods](#) for more information.

-> **attribute_back_fill@**

Sets chart item background fill attributes

Class ChartClass set

Format this.attribute_back_fill@(item, fg, bg)

Arguments

item	One of the following chart elements: default minor horizontal grid major horizontal grid major vertical grid minor vertical grid line, axis x<number> line, axis y<number> label, axis x<number> label, axis y<number> tik label, axis x<number> tik label, axis y<number> data label data title subtitle footer legend box legend title legend labels
fg	The name of color in colormap for foreground color, or an array of color names. If an array is passed, fg[0] is the name of the foreground color, fg[1] is the name of the background color.
bg	The name of color in colormap for background color.

Description The set method `attribute_back_fill@` sets the background fill attributes for the given chart component.

See [ChartClass Methods](#) for more information.

-> **attribute_character@**

Sets chart item character attributes

Class ChartClass set

Format this.attribute_character@(item, face, size, bold, italic)

Arguments

item	One of the following chart elements: default minor horizontal grid major horizontal grid major vertical grid minor vertical grid line, axis x<number> line, axis y<number> label, axis x<number> label, axis y<number> tik label, axis x<number> tik label, axis y<number> data label data title subtitle footer legend box legend title legend labels
face	The character font name, or an array of 4 values. If an array is passed, the array contains the following values: face[0] The font name. face[1] The font point size. face[2] The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE. face[3] The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

Description The set method `attribute_character@` sets the character attributes for the given chart component.

See [ChartClass Methods](#) for more information.

-> `attribute_field@`

Sets chart item alignment

Class ChartClass set

Format `this.attribute_field@(item, hAlign, vAlign)`

Arguments `item` One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid
- minor vertical grid
- line, axis x<number>
- line, axis y<number>
- label, axis x<number>
- label, axis y<number>
- tik label, axis x<number>
- tik label, axis y<number>
- data
- label
- data
- title
- subtitle
- footer
- legend box
- legend title
- legend labels

`hAlign` The horizontal alignment, or an array of 2 values. If an array is passed, the array contains the following values:

`hAlign[0]` The horizontal alignment.

`hAlign[1]` The vertical alignment.

`vAlign` The vertical alignment.

The valid alignment values are:

0	Left
1	Center
2	Right

Description The set method `attribute_field@` sets the alignment for the given chart component. See [ChartClass Methods](#) for more information.

-> `attribute_line_fill@`

Sets chart item line fill attributes

Class ChartClass set

Format `this.attribute_line_fill@(item, fg, bg)`

Arguments

item	One of the following chart elements: default minor horizontal grid major horizontal grid major vertical grid minor vertical grid line, axis x<number> line, axis y<number> label, axis x<number> label, axis y<number> tik label, axis x<number> tik label, axis y<number> data label data title subtitle footer legend box legend title legend labels
fg	The name of color in colormap for foreground color, or an array of color names. If an array is passed, <code>fg[0]</code> is the name of the foreground color, <code>fg[1]</code> is the name of the background color.

bg The name of color in colormap for background color.

Description The set method `attribute_line_fill@` sets the line fill attributes for the given chart component.

See [ChartClass Methods](#) for more information.

-> `attribute_line_style@`

Sets chart item line style attributes

Class ChartClass set

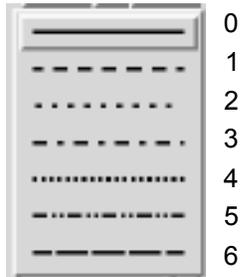
Format `this.attribute_line_style@(item, style, weight, first, join, final)`

Arguments item One of the following chart elements:

- default
- minor horizontal grid
- major horizontal grid
- major vertical grid
- minor vertical grid
- line, axis x<number>
- line, axis y<number>
- label, axis x<number>
- label, axis y<number>
- tik label, axis x<number>
- tik label, axis y<number>
- data
- label
- data
- title
- subtitle
- footer
- legend box
- legend title
- legend labels

style The line style, or an array of 5 values. If an array is passed, the array contains the following values:

style[1] The line style. Use one of the following numeric values:



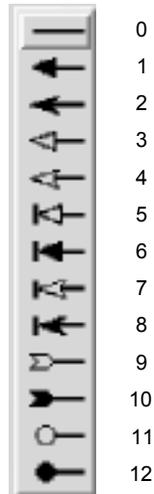
- style[1] The line width, in pixels
- style[2] The symbol to be used at a line's beginning.
- style[3] The symbol to be used when lines are joined.
- style[4] The symbol used at the end of a line.

weight

The line width, in pixels.

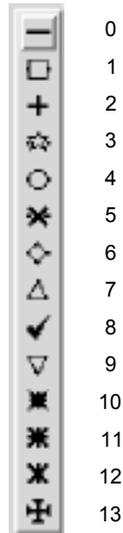
first

The symbol to be used at a line's beginning. Use one of the following numeric values:



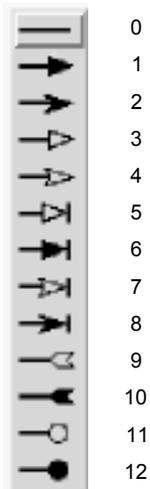
join

The symbol to be used when lines are joined. Use one of the following numeric values:



final

The symbol used at the end of a line. Use one of the following numeric values:



Description The set method `attribute_line_style@` sets the line style attributes for the given chart component.

See [ChartClass Methods](#) for more information.

-> attribute_shadow@

Sets chart item shadow attributes

Class ChartClass set

Format this.attribute_shadow@(item, type, color)

Arguments

item	One of the following chart elements: default minor horizontal grid major horizontal grid major vertical grid minor vertical grid line, axis x<number> line, axis y<number> label, axis x<number> label, axis y<number> tik label, axis x<number> tik label, axis y<number> data label data title subtitle footer legend box legend title legend labels
type	The shadow type, or an array of 2 values. If an array is passed, the array contains the following values: type[0] The shadow type. The valid types are: 0 none 1 background 2 local type[1] The shadow color
color	The name of the color in the colormap.

Description The set method attribute_shadow@ sets the shadow attributes for the given chart component.

See [ChartClass Methods](#) for more information.

-> axis@

Sets chart axis definition

Class ChartClass set

Format this.axis@(name, format chart_axis_info@ info)

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
info	The axis information being set.

Description The set method axis@ sets the chart axis definition in chart_axis_info@ format. The chart_axis_info@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

-> axis_auto_tik_flags@

Sets axis auto tik settings

Class ChartClass set

Format this.axis_auto_tik_flags@(name, major, minor)

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
major	A Boolean value, or an array of 2 Boolean values. A single Boolean value, if set to TRUE means automatically set major tik value. An array of Boolean values sets the major and minor tik values.

minor A Boolean value, if set to TRUE means automatically set minor tik value.

Description The set method `axis_auto_tik_flags@` sets the axis auto tik settings.

See [ChartClass Methods](#) for more information.

axis_auto_value_flags@

Sets axis auto value settings

Class ChartClass set

Format `this.axis_auto_value_flags@(name, max, min, base)`

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
max	A Boolean value, or an array of 3 Boolean values. A single Boolean value, if set to TRUE means automatically set maximum axis value. An array of Boolean values sets the max, min, and base axis values.
min	A Boolean value, if set to TRUE means automatically set minimum axis value.
base	A Boolean value, if set to TRUE means automatically set base axis value.

Description The set method `axis_auto_value_flags@` sets the axis auto value settings.

See [ChartClass Methods](#) for more information.

-> axis_bar_spacing@

Sets axis bar spacing

Class ChartClass set

Format `this.axis_bar_spacing@(name, margin, overlap)`

Arguments name The name of one of the chart's axes. This name takes the following form:

	axis x1
	axis x2
	...
	axis y1
	axis y2
	...
margin	The bar margin, or an array of 2 values. If an array is passed, margin[0] is the bar margin value, margin[1] is the bar overlap.
overlap	The bar overlap.

Description The set method `axis_bar_spacing@` sets the axis auto value settings.

See [ChartClass Methods](#) for more information.

-> `axis_label@`

Sets axis label info

Class ChartClass set

Format `this.axis_label@(name, label, xmargin, ymargin, align, tight)`

Arguments name The name of one of the chart's axes. This name takes the following form:

```
axis x1
axis x2
...
axis y1
axis y2
...
```

label The axis label string, or an array of 5 values. If an array is passed, the array contains the following values:

```
label[0] The label string.
label[1] The label x-axis margin.
label[2] The label y-axis margin.
label[3] The label alignment:
          0 Left
          1 Center
          2 Right
label[4] The tight label setting.
```

xmargin	The label x-axis margin, in pixels.
ymargin	The label y-axis margin, in pixels.
align	The label alignment:
	0 Left
	1 Center
	2 Right
tight	The spacing between labels and tik marks.

Description The set method `axis_label@` sets the axis label information. See [ChartClass Methods](#) for more information.

-> axis_labels@

Sets chart axis label definition

Class ChartClass set

Format `this.axis_labels@(name, format chart_axis_labels@ info)`

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
info	The axis label information.

Description The set method `axis_labels@` sets the chart axis label definition in `chart_axis_labels@` format. The `chart_axis_labels@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

-> axis_line@

Sets chart axis line attributes

Class ChartClass set

Format this.axis_line@(name, format chart_axis@ info)

Arguments name The name of one of the chart's axes. This name takes the following form:
 axis x1
 axis x2
 ...
 axis y1
 axis y2
 ...
info Information describing one of a chart's axis.

Description The set method axis_line@ sets the chart axis line attributes in chart_axis@ format. The chart_axis@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

-> axis_position@

Sets axis position info

Class ChartClass set

Format this.axis_position@(name, pos, value, behind, hidden)

Arguments name The name of one of the chart's axes. This name takes the following form:
 axis x1
 axis x2
 ...
 axis y1
 axis y2
 ...
pos The axis position, or an array of 4 values. If an array is passed, the array contains the following values:
 pos[0] The axis position.
 left 0
 right 1
 bottom 2
 top 3
 v_val 4
 h_val 5

	pos[1]	The axis value.
	pos[2]	Boolean value, TRUE means axis appears behind data.
	pos[3]	Boolean value, TRUE means axis is hidden.
value		The axis value, one of the following strings: "fixed" "fixed, primary" "fixed, secondary" "floating"
behind		Boolean value, TRUE means axis appears behind data.
hidden		Boolean value, TRUE means axis is hidden.

Description The set method `axis_position@` sets the axis position information. See [ChartClass Methods](#) for more information.

<- axis_labels@

Returns chart axis label definition

Class ChartClass get

Format `format chart_axis_labels@ info = this.axis_labels@(name)`

Arguments name The name of one of the chart's axes. This name takes the following form:

```
axis x1
axis x2
...
axis y1
axis y2
...
```

Description The get method `axis_labels@` returns the chart axis label definition in `chart_axis_labels@` format. The `chart_axis_labels@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

<- axis_line@

Returns chart axis line attributes

Class ChartClass get

Format format chart_axis@ info = this.axis_line@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:

- axis x1
- axis x2
- ...
- axis y1
- axis y2
- ...

Description The get method axis_line@ returns the chart axis line attributes in chart_axis@ format. The chart_axis@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

<- axis_position@

Returns chart axis position info

Class ChartClass get

Format valArray = this.axis_position@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:

- axis x1
- axis x2
- ...
- axis y1
- axis y2
- ...

Description The get method axis_position@ returns the chart axis position information as an array of values. The array contains the following information:

valArray[0] The axis position.

valArray[1] The axis value.
valArray[2] Boolean value, TRUE means axis appears behind data.
valArray[3] Boolean value, TRUE means axis is hidden.

See [ChartClass Methods](#) for more information.

<- axis_tik@

Returns chart axis tik info

Class ChartClass get

Format valArray = this.axis_tik@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:
axis x1
axis x2
...
axis y1
axis y2
...

Description The get method axis_tik@ returns the chart axis tik information as an array of values. The array contains the following information:

valArray[0] Major tik type.
valArray[1] Major tik size.
valArray[2] Quantity of major tiks.
valArray[3] Minor tik type.
valArray[4] Minor tik size.
valArray[5] Quantity of minor tiks.

See [ChartClass Methods](#) for more information.

<- axis_tik_label_info@

Returns chart axis tik label info

Class ChartClass get

Format valArray = this.axis_tik_label_info@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:
 axis x1
 axis x2
 ...
 axis y1
 axis y2
 ...

Description The get method axis_tik_label_info@ returns the chart axis tik label information as an array of values. The array contains the following information:

valArray[0] The tik alignment.

valArray[1] The tik margin.

valArray[2] The spacing between labels and tik marks.

See [ChartClass Methods](#) for more information.

<- axis_values@

Returns chart axis value info

Class ChartClass get

Format valArray = this.axis_values@(name)

Arguments name The name of one of the chart's axes. This name takes the following form:
 axis x1
 axis x2
 ...
 axis y1
 axis y2
 ...

Description The get method axis_values@ returns the chart axis value information as an array of values. The array contains the following information:

valArray[0] The maximum axis value.

valArray[1] The minimum axis value.

valArray[2] The base axis value.

The setting is ignored if the corresponding auto value is set to TRUE. See [axis auto value flags@](#) for information on the auto value settings.

See [ChartClass Methods](#) for more information.

<- chart_3d_info@

Returns chart 3D info

Class ChartClass get

Format valArray = this.chart_3d_info@

Description The get method `chart_3d_info@` returns the chart 3D information as an array of values. The array contains the following information:

valArray[0] A Boolean value, TRUE means 3D effects are enabled.

valArray[1] The chart pitch, in degrees(-90 to 90).

valArray[2] The chart projection. In effect when right angle projection is FALSE.

valArray[3] The depth, in pixels.

valArray[4] A Boolean value, TRUE means right angle projection.

See [ChartClass Methods](#) for more information.

<- clear_mode@

Returns chart clear mode

Class ChartClass get

Format mode = this.clear_mode@

Description The get method `clear_mode@` returns the mode used to set the chart area color. The valid clear modes are:

none Do not use a color for the chart area

widget Use the widget color in the chart area

window Use the window color in the chart area

The widget and window colors are set with the control panel and work area color settings.

See [ChartClass Methods](#) for more information.

<- data@

Returns chart data

Class ChartClass get

Format dataArray = this.data@

Description The get method data@ returns the data used to create the chart as an array of numeric information.

See [ChartClass Methods](#) for more information.

<- data_point_label_info@

Returns point label information

Class ChartClass get

Format valArray = this.data_point_label_info@(name, point)

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2

...

However, the value following data can be any user-defined string.

point The point in the data group.

Description The get method data_point_label_info@ returns the point label information as an array of values. The array contains the following information:

valArray[0] Label x-offset.

valArray[1] Label y-offset.

valArray[2] Label string.

valArray[3] Point display type.

See [ChartClass Methods](#) for more information.

<- decorations@

Returns border and grid information

Class ChartClass get

Format format chart_decorations@ info = this.decorations@

Description The get method decorations@ returns the border and grid information in chart_decorations@ format. The chart_decorations@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format. The format contains the following 8 Boolean chart properties:

```
format chart_decorations@
  close_top,
  close_bottom,
  close_left,
  close_right,
  major_h_grids,      ' horizontal
  minor_h_grids,
  major_v_grids,      ' vertical
  minor_v_grids
```

The Boolean value indicates if the border or grid is displayed.

See [ChartClass Methods](#) for more information.

<- effects_3d@

Returns general 3D information: yaw, pitch, projection, and depth

Class ChartClass get

Format format chart_3D_effect@ info = this.effects_3d@

Description The get method effects_3d@ returns the chart 3D information in chart_3D_effect@ format. The chart_3D_effect@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format. The format is defined as:

```
format chart_3D_effect@
  enabled,
  yaw,
  pitch,
  projection,
```

depth

See [ChartClass Methods](#) for more information.

<- group@

Returns group information

Class ChartClass get

Format format chart_group@ group = this.group@(name)

Arguments name The name of the group being created. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2

...

However, the value following data can be any user-defined string.

Description The get method group@ returns the group information for the passed group name in chart_group@ format. The chart_group@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

<- group_axes@

Returns group axes

Class ChartClass get

Format valArray = this.group_axes@(name)

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2

...

However, the value following data can be any user-defined string.

Description The get method `group_axes@` returns the group axes as an array of values. The array contains the following information:

`valArray[0]` x-axis name.

`valArray[1]` y-axis name.

`valArray[2]` z-axis name.

See [ChartClass Methods](#) for more information.

`<- group_create@`

Adds a group to a chart

Class ChartClass get

Format `name = this.group_create@(name, format chart_group@ group)`

Arguments `name` The name of the group being created. If this argument is NULL or is not a string, a name is created for you. A group name created by Graphics is of the following form:

`data 0`

`data 1`

`data 2`

...

However, the value following data can be any user-defined string.

`group` The data for the group, in `chart_group@` format. The `chart_group@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format.

Description The get method `group_create@` adds a group to the chart and returns the group name.

See [ChartClass Methods](#) for more information.

`<- group_create_from_data@`

Adds a group to a chart

Class ChartClass get

Format `name = this.group_create_from_data@(data, type)`

Arguments `data` An array of chart data.

type A string indicating the group type.

Description The get method `group_create_from_data@` adds a group to the chart and returns the group name.

See [ChartClass Methods](#) for more information.

<- group_destroy@

Removes a group

Class ChartClass get

Format `groupArray = this.group_destroy@(name)`

Arguments name The name of the group being removed. A group name created by Graphics is of the following form:

data 0
data 1
data 2

...

However, the value following data can be any user-defined string.

Description The get method `group_destroy@` removes a group from the chart and returns the remaining group names as an array of strings.

See [ChartClass Methods](#) for more information.

<- group_label@

Returns group label information

Class ChartClass get

Format `valArray = this.group_label@(name)`

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2

...

However, the value following data can be any user-defined string.

Description The get method `group_label@` returns the group label information as an array of values. The array contains the following information:

`valArray[0]` Label type.
`valArray[1]` Label alignment.
`valArray[2]` Label x-axis offset.
`valArray[3]` Label y-axis offset.

See [ChartClass Methods](#) for more information.

`<- group_legend@`

Returns group legend

Class ChartClass get

Format `legend = this.group_legend@(name)`

Arguments `name` The name of the data group. A name created by Applixware Graphics is in the following form:

`data 0`
`data 1`
`data 2`
...

However, the value following data can be any user-defined string.

Description The get method `group_legend@` returns the group label information as a string.

See [ChartClass Methods](#) for more information.

`<- group_names@`

Returns names of all groups

Class ChartClass get

Format `groupArray = this.group_names@`

Description The get method `group_names@` returns all group names in the chart as an array of strings.

See [ChartClass Methods](#) for more information.

<- is_3d_on@

Returns chart 3D status

Class ChartClass get

Format flag = this.is_3d_on@

Description The get method `is_3d_on@` returns the chart 3D status as a Boolean value. The method returns TRUE if 3D effects are enabled, otherwise the method returns FALSE.

See [ChartClass Methods](#) for more information.

<- is_data_series_column_ordered@

Returns data series ordering

Class ChartClass get

Format flag = this.is_data_series_column_ordered@

Description The get method `is_data_series_column_ordered@` returns the data series ordering as a Boolean value. The method returns TRUE if the data series is column-ordered. The method returns FALSE if the data series is row-ordered.

A chart data series is a two-dimensional array. If a data series is row-ordered, `data[0,x]` is one data group, `data[1,x]` is the next data group, and so on. If a data series is column-ordered, `data[x,0]` is one data group, `data[x,1]` is the next data group, and so on. Use the set method [data@](#) to set the chart data.

See [ChartClass Methods](#) for more information.

<- is_display_on_update@

Returns chart display mode

Class ChartClass get

Format flag = this.is_display_on_update@

Description The get method `is_display_on_update@` returns the chart display mode as a Boolean value. The method returns TRUE if chart is redrawn after every change. The method returns FALSE if the chart must be redrawn with the set method `display_chart@`.

See [ChartClass Methods](#) for more information.

<- is_first_col_for_legends@

Returns use of first data column

Class ChartClass get

Format flag = this.is_first_col_for_legends@

Description The get method `is_first_col_for_legends@` returns the use of the first data column as a Boolean value. A chart data series is a two-dimensional array. The method returns TRUE if the first column of chart data, `data[x,0]`, is used as data group legend titles. The method returns FALSE if the first column of chart data is used to draw the chart. Use the set method `data@` to set the chart data.

See [ChartClass Methods](#) for more information.

<- is_first_row_for_labels@

Returns use of first data row

Class ChartClass get

Format flag = this.is_first_row_for_labels@

Description The get method `is_first_row_for_labels@` returns the use of the first data row as a Boolean value. A chart data series is a two-dimensional array. The method returns TRUE if the first row of chart data, `data[0,x]`, is used as chart tik labels. The method returns FALSE if the first row of chart data is used to draw the chart. Use the set method `data@` to set the chart data.

See [ChartClass Methods](#) for more information.

<- legend@

Returns legend information

Class ChartClass get

Format format chart_legend@ legend = this.legend@

Description The get method legend@ returns the chart legend information in chart_legend@ format. The chart_legend@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

<- legend_attr@

Returns legend information

Class ChartClass get

Format valArray = this.legend_attr@

Description The get method legend_attr@ returns the chart legend information as an array of values. The array contains the following information:

valArray[0] A Boolean value, TRUE means arrange legends by row.

valArray[1] The maximum quantity of legends per row or column.

valArray[2] A Boolean value, TRUE means place group name before sample.

valArray[3] A Boolean value, TRUE means legend uses available space x-axis space in the chart.

valArray[4] A Boolean value, TRUE means legend uses available space y-axis space in the chart.

See [ChartClass Methods](#) for more information.

<- legend_box@

Returns legend position information

Class ChartClass get

Format valArray = this.legend_box@

Description The get method legend_box@ returns the chart legend position information as an array of values. The array contains the following information:

valArray[0] Horizontal alignment.

"left"

"center"

"right"

valArray[1] Vertical alignment.

"top"

"middle"

"bottom"

valArray[2] Horizontal offset.

valArray[3] Vertical offset.

See [ChartClass Methods](#) for more information.

<- legend_display@

Returns legend display status

Class ChartClass get

Format flag = this.legend_display@

Description The get method legend_display@ returns the legend display status as a Boolean value. The method returns TRUE if the legend is displayed, otherwise the method returns FALSE.

See [ChartClass Methods](#) for more information.

<- legend_title@

Returns legend title information

Class ChartClass get

Format valArray = this.legend_title@

Description The get method `legend_title@` returns the chart legend title information as an array of values. The array contains the following information:

<code>valArray[0]</code>	The title string.
<code>valArray[1]</code>	The title alignment.
0	Left
1	Center
2	Right

See [ChartClass Methods](#) for more information.

`<- margins@`

Returns chart margins

Class ChartClass get

Format `marginArray = this.margins@`

Description The get method `margins@` returns the margin within the chart's extent. The margins are within the area used to draw the chart.. The returned array has four numeric elements representing, in order, the:

- Left margin
- Top margin
- Right margin
- Bottom margin

Note that the title, subtitle, footer, and legend ignore these margins and are placed on the extent.

See [ChartClass Methods](#) for more information.

`<- null_format@`

Indicates how NULL items are displayed

Class ChartClass get

Format `formatString = this.null_format@`

Description The get method `null_format@` returns a string indicating how NULL data points are displayed. The returned value is one of the following strings:

- zero, indicating that null points are displayed as zeroes.
- span, indicating that the point is ignored (the graph *spans* this point).
- gap, indicating that if a line is being drawn, a gap is displayed where the point should be.

See [ChartClass Methods](#) for more information.

<- orientation@

Returns chart's orientation

Class ChartClass get

Format formatString = this.orientation@

Description The get method `null_format@` returns a chart's orientation, the direction of the chart's major axis, as one of the following strings:

- vertical
- horizontal

See [ChartClass Methods](#) for more information.

<- proportional@

Returns proportional state

Class ChartClass get

Format flag = this.proportional@

Description The get method `proportional@` returns a Boolean value for the proportional state used when scaling the chart in the chart area. TRUE means make the chart proportional when scaling.

See [ChartClass Methods](#) for more information.

<- scale_mode@

Returns scaling mode

Class ChartClass get

Format mode = this.scale_mode@

Description The get method `scale_mode@` returns the mode used to scale the chart in the chart area. The valid scaling modes are:

clip to fit Draw chart at full size, cutting off parts that do not fit the chart area

scale to fit Scale the chart to fit the chart area

scale if clips Scale the chart to fit the chart area if necessary, otherwise draw at full size

See [ChartClass Methods](#) for more information.

<- titles@

Returns all of a chart's title information

Class ChartClass get

Format format chart_titles@ info = this.titles@

Description The get method `titles@` returns the text and alignment for a chart's title, subtitle, and footer, in `chart_legend@` format. The `chart_legend@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format. The format is defined as:

```
format chart_title@
  title,
  subtitle,
  footer,
  title_alignment,
    0    left
    1    center
    2    right
  subtitle_alignment,
    0    left
    1    center
    2    right
```

footer_alignment
0 left
1 center
2 right

See [ChartClass Methods](#) for more information.

<- title_component@

Returns title component information

Class ChartClass get

Format valArray = this.title_component@(item)

Arguments item One of the following title components:
title
subtitle
footer

Description The get method title_component@ returns the title component information as an array of values. The array contains the following information:

valArray[0] Item string
valArray[1] Item alignment.
valArray[2] A Boolean value, TRUE means expand margin for item text.
valArray[3] Item x-axis offset.
valArray[3] Item y-axis offset.

See [ChartClass Methods](#) for more information.

<- type@

Returns a chart's type

Class ChartClass get

Format formatString = this.type@

Description The get method type@ returns a string indicating a chart's type. See [Chart Types](#) for a list of defined chart types.

See [ChartClass Methods](#) for more information.

-> axis_tik@

Sets axis tik info

Class ChartClass set

Format this.axis_tik@(name, mtype, msize, mqty, ntype, nsize, nqty)

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
mtype	The major tik type, or an array of 6 values. If an array is passed, the array contains the following values: mtype[0] Major tik type. One of the following values: 0 None 1 Inside 2 Outside 3 Across mtype[1] Major tik size. mtype[2] Quantity of major tiks. mtype[3] Minor tik type. mtype[4] Minor tik size. mtype[5] Quantity of minor tiks.
msize	Major tik size.
mqty	Quantity of major tiks.

n _{type}	Minor tik type.
n _{size}	Minor tik size.
n _{qty}	Quantity of minor tiks.

Description The set method `axis_tik@` sets the axis tik information. See [ChartClass Methods](#) for more information.

-> `axis_tik_label_attrs@`

Sets axis tik label info

Class ChartClass set

Format `this.axis_tik_label_attrs@(name, align, margin, tight)`

Arguments	name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
	align	The tik alignment, or an array of 3 values. If an array is passed, the array contains the following values: align[0] The tik alignment. One of the following values: 0 Left 1 Center 2 Right 3 Justified align[1] The tik margin. align[2] The tight tik setting.
	margin	The tik margin.
	tight	The spacing between labels and tik marks.

Description The set method `axis_tik@` sets the axis tik label information. See [ChartClass Methods](#) for more information.

-> axis_values@

Sets chart axis value info

Class ChartClass set

Format this.axis_values@(name, max, min, base)

Arguments

name	The name of one of the chart's axes. This name takes the following form: axis x1 axis x2 ... axis y1 axis y2 ...
max	The maximum axis value, or an array of 3 values. If an array is passed, the array contains the following values: max[0] The maximum axis value. max[1] The minimum axis value. max[2] The base axis value.
min	The minimum axis value.
base	The base axis value.

Description The set method axis_values@ sets the chart axis value information.

The setting is ignored if the corresponding auto value is set to TRUE. See [axis_auto_value_flags@](#) for information on the auto value settings.

See [ChartClass Methods](#) for more information.

-> chart_3d_info@

Sets chart 3D info

Class ChartClass set

Format this.chart_3d_info@(enable, yaw, pitch, proj, depth, angle)

Arguments

enable	A Boolean value, TRUE means 3D effects are enabled.
yaw	The chart yaw, in degrees(-90 to 90).

pitch	The chart pitch, in degrees(-90 to 90).
proj	The chart projection. In effect when right angle projection is FALSE.
depth	The depth, in pixels.
angle	A Boolean value, TRUE means right angle projection.

Description The set method `chart_3d_info@` sets the chart 3D information.

See [ChartClass Methods](#) for more information.

-> `clear_mode@`

Sets chart clear mode

Class ChartClass set

Format `this.clear_mode@(mode)`

Arguments

mode	The color to use in the chart area. The valid clear modes are:
none	Do not use a color for the chart area
widget	Use the widget color in the chart area
window	Use the window color in the chart area

Description The set method `clear_mode@` set the mode used to set the chart area color. The widget and window colors are set with the control panel and work area color settings.

See [ChartClass Methods](#) for more information.

-> `copy_to_clipboard@`

Copies chart to Applixware clipboard

Class ChartClass set

Format `this.copy_to_clipboard@`

Description The set method `copy_to_clipboard@` copies the chart to the Applixware clipboard as an Applixware Graphics object. You can paste a chart in an Applixware Words, Graphics, or Spreadsheets document as an Applixware Graphics object. You can also paste the chart in WordsClass, SpreadsheetsClass, or GraphicsClass objects that exist in Applixware Builder applications.

See [ChartClass Methods](#) for more information.

-> create@

Creates a chart

Class ChartClass set

Format this.create@(type, data[, title[, subtitle[, footer[, x_title[, y_title[, legendFlag]]]]])

Arguments

type	A string indicating the chart type. See Chart Types for a list of defined chart types.
data	An array of chart data.
title	A string indicating the chart's title.
subtitle	A string indicating the chart's subtitle.
footer	A string indicating the chart's footer.
x_title	A string indicating the chart's x-axis title.
y_title	A string indicating the chart's y-axis title.
legendFlag	A Boolean value. TRUE displays the legend, FALSE does not display the legend.

Description The set method create@ creates a chart with the passed information. Use the set method display_chart@ to display the chart after creating the chart.

See [ChartClass Methods](#) for more information.

-> data@

Sets chart data

Class ChartClass set

Format this.data@(dataArray)

Arguments dataArray An array of numeric information.

Description The set method data@ sets the data used to create the chart.

See [ChartClass Methods](#) for more information.

-> data_point_label_info@

Sets point label info

Class ChartClass set

Format this.data_point_label_info@(name, point, type, xOff, yOff, string)

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

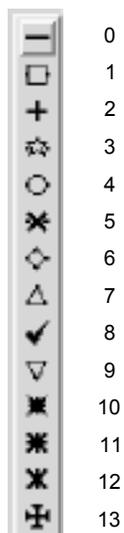
data 0
data 1
data 2
...

However, the value following data can be any user-defined string.

point A point in the data group, or an array of 5 values. If an array is passed, the array contains the following values:

point[0] A point in the data group.
point[1] Point display type.
point[2] Label x-offset.
point[3] Label y-offset.
point[4] Label string.

type Point display type. One of the following numeric values:



xOff	Label x-offset , in pixels.
yOff	Label y-offset, in pixels.
string	Label string.

Description The set method `data_point_label_info@` sets the point label information.
See [ChartClass Methods](#) for more information.

-> decorations@

Sets border and grid information

Class ChartClass set

Format `this.decorations@(format chart_decorations@ inf)`

Arguments int Border and grid information.

Description The set method `decorations@` sets the border and grid information in `chart_decorations@` format. The `chart_decorations@` format is defined in the `/install_dir/axdata/elf/chart_.am` file. Include this file in any sources using the format. The format contains the following 8 Boolean chart properties:

```
format chart_decorations@
    close_top,
    close_bottom,
    close_left,
    close_right,
    major_h_grids,      ' horizontal
    minor_h_grids,
    major_v_grids,     ' vertical
    minor_v_grids
```

The Boolean value indicates if the border or grid is displayed.

See [ChartClass Methods](#) for more information.

-> destroy@

Removes a chart

Class ChartClass set

Format this.destroy@

Description The set method destroy@ removes a chart and frees the memory associated with the chart.

See [ChartClass Methods](#) for more information.

-> display_chart@

Displays a chart

Class ChartClass set

Format this.display_chart@

Description The set method display_chart@ displays the chart if it is not already displayed. Use this method after setting creating the chart and setting the chart information.

See [ChartClass Methods](#) for more information.

-> drawing_area@

Sets chart position and dimensions

Class ChartClass set

Format this.drawing_area@(x, y, width, height)

Arguments	x	The x-position of the chart, in pixels, from the upper left corner of the drawable area.
	y	The y-position of the chart, in pixels, from the upper left corner of the drawable area.
	width	The chart width, in pixels.
	height	The chart height, in pixels.

Description The set method drawing_area@ sets the chart position and dimensions in the drawable chart area.

See [ChartClass Methods](#) for more information.

-> edit_3d_dlg@

Displays a 3D effect dialog box

Class ChartClass set

Format this.edit_3d_dlg@

Description The set method edit_3d_dlg@ displays the 3D Effect of Chart dialog box. Use the dialog box to set the chart 3D attributes.

See [ChartClass Methods](#) for more information.

-> edit_axes_dlg@

Displays an axes dialog box

Class ChartClass set

Format this.edit_axes_dlg@

Description The set method edit_axes_dlg@ displays the Axes dialog box. Use the dialog box to set the chart axes attributes.

See [ChartClass Methods](#) for more information.

-> edit_frame_and_grid_dlg@

Displays a frame and grid dialog box

Class ChartClass set

Format this.edit_frame_and_grid_dlg@

Description The set method edit_frame_and_grid_dlg@ displays the Frame and Grids of Chart dialog box. Use the dialog box to set the chart frame and grid settings.

See [ChartClass Methods](#) for more information.

-> edit_layout_dlg@

Displays a layout dialog box

Class ChartClass set

Format this.edit_layout_dlg@

Description The set method edit_layout_dlg@ displays the Layout of Chart dialog box. Use the dialog box to set the chart layout settings.

See [ChartClass Methods](#) for more information.

-> edit_legend_dlg@

Displays a legend dialog box

Class ChartClass set

Format this.edit_legend_dlg@

Description The set method edit_legend_dlg@ displays the Legend dialog box. Use the dialog box to set the chart legend attributes.

See [ChartClass Methods](#) for more information.

-> edit_titles_dlg@

Displays a title dialog box

Class ChartClass set

Format this.edit_titles_dlg@

Description The set method edit_titles_dlg@ displays the Titles of Chart dialog box. Use the dialog box to set the chart title settings.

See [ChartClass Methods](#) for more information.

-> effects_3d@

Sets general 3D information: yaw, pitch, projection, and depth

Class ChartClass set

Format this.effects_3d@(format chart_3D_effect@ info)

Arguments info Chart 3D information

Description The set method effects_3d@ sets the chart 3D information in chart_3D_effect@ format. The chart_3D_effect@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format. The format is defined as:

```
format chart_3D_effect@
    enabled,
    yaw,
    pitch,
    projection,
    depth
```

See [ChartClass Methods](#) for more information.

-> enable_3d@

Enables 3D chart effects

Class ChartClass set

Format this.enable_3d@(flag)

Arguments flag A Boolean value. TRUE enables 3D effects, FALSE disables 3D effects.

Description The set method enable_3d@ sets the chart 3D effects. Use the set method effects_3d@ to set the chart 3D information. Use the get method is_3d_on@ to determine if chart 3D effects are enabled.

See [ChartClass Methods](#) for more information.

-> group@

Sets group information

Class ChartClass set

Format this.group@(name, format chart_group@ group)

Arguments name The name of the group being created. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2
...

However, the value following data can be any user-defined string.

group The data for the group.

Description The set method group@ sets the group information for the passed group name in chart_group@ format. The chart_group@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

-> group_axes@

Sets group axes

Class ChartClass set

Format this.group_axes@(name, x, y, z)

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

data 0
data 1
data 2
...

However, the value following data can be any user-defined string.

x The x-axis, or an array of 3 values. If an array is passed, the array contains the following values:

	x[0]	x-axis name.
	x[1]	y-axis name.
	x[2]	z-axis name.
y		y-axis name.
z		z-axis name.

Description The set method `group_axes@` sets the group axes
See [ChartClass Methods](#) for more information.

-> `group_label@`

Sets group label info

Class ChartClass set

Format `this.group_label@(name, type, align, xOff, yOff)`

Arguments	name	The name of the data group. A name created by Applixware Graphics is in the following form: data 0 data 1 data 2 ...
	type	However, the value following data can be any user-defined string. The label type, or an array of 4 values. If an array is passed, the array contains the following values: type[0] Label type. 0 none 1 value 2 percent 3 string type[1] Label alignment. type[2] Label x-axis offset. type[3] Label y-axis offset.
	align	Label alignment.
	xOff	Label x-axis offset.
	yOff	Label y-axis offset.

Description The set method `group_label@` sets the group label information.

See [ChartClass Methods](#) for more information.

-> `group_legend@`

Sets group legend

Class ChartClass set

Format `this.group_legend@(name, legend)`

Arguments name The name of the data group. A name created by Applixware Graphics is in the following form:

data 0

data 1

data 2

...

However, the value following data can be any user-defined string.

legend The group legend string.

Description The set method `group_legend@` sets the group legend.

See [ChartClass Methods](#) for more information.

-> `is_data_series_column_ordered@`

Sets data series ordering

Class ChartClass set

Format `this.is_data_series_column_ordered@(flag)`

Arguments flag A Boolean value. TRUE uses the data series in column order, FALSE uses the data series in row order. The default ordering is row order.

Description The set method `is_data_series_column_ordered@` sets the data series ordering. A chart data series is a two-dimensional array. If a data series is row-ordered, `data[0,x]` is one data group, `data[1,x]` is the next data group, and so on. If a data series is column-ordered, `data[x,0]` is one data group, `data[x,1]` is the next data group, and so on. Use the set method [data@](#) to set the chart data.

The settings for this method effect how labels and data legends are created when the methods [is first row for labels@](#) and [is first col for legends@](#) are set to TRUE. If this method is not set, or set to FALSE, labels are created with the first row of information, and legends are created with the first column of information. If this method is set to TRUE the labels are created with the first column of information and legends are created with the first row of information.

See [ChartClass Methods](#) for more information.

-> [is_display_on_update@](#)

Sets chart display mode

Class ChartClass set

Format this.is_display_on_update@(flag)

Arguments flag A Boolean value. TRUE redraws the chart after every change, FALSE does not redraw the chart after a change.

Description The set method [is_display_on_update@](#) sets the chart display mode. If [is_display_on_update@](#) is set to FALSE the chart must be redrawn with the set method [display_chart@](#).

See [ChartClass Methods](#) for more information.

-> [is_first_col_for_legends@](#)

Determines if first column of data is for legend titles

Class ChartClass set

Format this.is_first_col_for_legends@(flag)

Arguments flag A Boolean value. TRUE uses the first column of chart data as data group legend titles, FALSE uses the first column of chart data to draw the chart. The default action is to use the first column of chart data to draw the chart.

Description The set method [is_first_col_for_legends@](#) determines if the first column of chart data is for data group legend titles. A chart data series is a two-dimensional array. If the method is set to TRUE the first column of data, `data[x,0]`, is used as data group legend titles. If the [is data series column ordered@](#) method is set to TRUE the first row of

data, data[0, x], is used as data group legend titles. Use the set method [data@](#) to set the chart data.

See [ChartClass Methods](#) for more information.

-> [is_first_row_for_labels@](#)

Determines if first row of data is for chart labels

Class ChartClass set

Format this.is_first_row_for_labels@(flag)

Arguments flag A Boolean value. TRUE uses the first row of chart data as chart tik labels, FALSE uses the first row of chart data to draw the chart. The default action is to use the first row of chart data to draw the chart.

Description The set method [is_first_row_for_labels@](#) determines if the first row of chart data is for chart tik labels. A chart data series is a two-dimensional array. If the method is set to TRUE the first row of data, data[0,x], is used as chart tik labels. If the [is_data_series_column_ordered@](#) method is set to TRUE the first coulmn of data, data[x, 0], is used as chart tik labels. Use the set method [data@](#) to set the chart data.

See [ChartClass Methods](#) for more information.

-> [legend@](#)

Sets legend information

Class ChartClass set

Format this.legend@(format chart_legend@ legend)

Arguments legend The legend information.

Description The set method [legend@](#) sets the chart legend information in [chart_legend@](#) format. The [chart_legend@](#) format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format.

See [ChartClass Methods](#) for more information.

-> legend_attr@

Sets legend information

Class ChartClass set

Format this.legend_attr@(byFlag, rowQty, bFlag, xSpace, ySpace)

Arguments

byFlag	A Boolean value, TRUE means arrange legends by row, or an array of 5 values. If an array is passed, the array contains the following values: byFlag[0] A Boolean value, TRUE means arrange legends by row. byFlag[1] The maximum quantity of legends per row or column. byFlag[2] A Boolean value, TRUE means place group name before sample. byFlag[3] A Boolean value, TRUE means legend uses available space x-axis space in the chart. byFlag[4] A Boolean value, TRUE means legend uses available space y-axis space in the chart.
rowQty	The maximum quantity of legends per row or column.
bFlag	A Boolean value, TRUE means place group name before sample.
xSpace	A Boolean value, TRUE means legend uses available space x-axis space in the chart.
ySpace	A Boolean value, TRUE means legend uses available space y-axis space in the chart.

Description The set method legend_attr@ sets the chart legend information.

See [ChartClass Methods](#) for more information.

-> legend_box@

Sets legend position information

Class ChartClass set

Format this.legend_box@(hAlign, vAlign, hOff, vOff)

Arguments

hAlign	The horizontal alignment, or an array of 4 values. If an array is passed, the array contains the following values:
--------	--

	hAlign[0]	Horizontal alignment. One of the following strings: "left" "center" "right"
	hAlign[1]	Vertical alignment.
	hAlign[2]	Horizontal offset.
	hAlign[3]	Vertical offset.
vAlign		Vertical alignment.. One of the following strings: "top" "middle" "bottom"
	hOff	Horizontal offset.
	vOff	Vertical offset.

Description The set method `legend_box@` sets the chart legend position information. See [ChartClass Methods](#) for more information.

-> legend_display@

Sets legend display status

Class ChartClass set

Format `this.legend_display@(flag)`

Arguments `flag` A Boolean value. TRUE displays the legend, FALSE hides the legend.

Description The set method `legend_display@` sets the legend display status.

See [ChartClass Methods](#) for more information.

-> legend_title@

Sets legend title information

Class ChartClass set

Format this.legend_title@(title, align)

Arguments title The legend title string, or an array of 2 values. If an array is passed, the array contains the following values:

- title[0] The title string.
- title[1] The title alignment.

align The title alignment in the legend box. One of the following values:

- 0 Left
- 1 Center
- 2 Right

Description The set method legend_title@ sets the chart legend title information.

See [ChartClass Methods](#) for more information.

-> margins@

Sets the margin within the chart area

Class ChartClass set

Format this.margins@(flag)

Arguments x1 The left margin.
y1 The chart's top margin.
x2 The chart's right margin.
y2 The chart's bottom margin

Description The set method margins@ sets the margin within the chart's extent. The margins are within the area used to draw the chart. The default value is 500 mills. (1000 mills = 1 inch)

Note that the title, subtitle, footer, and legend ignore these margins and are placed on the extent.

See [ChartClass Methods](#) for more information.

-> null_format@

Sets format for displaying NULL points

Class ChartClass set

Format this.null_format@(formatString)

Arguments formatString One of the following values:

- zero NULL points are displayed as zeroes.
- span The point is ignored (the graph *spans* this point).
- gap A gap is displayed in where the point should be.

Description The set method null_format@ sets the format for displaying NULL points.

See [ChartClass Methods](#) for more information.

-> orientation@

Sets the chart's orientation to horizontal or vertical

Class ChartClass set

Format this.orientation@(type)

Arguments type The chart's orientation, one of the following values:

- vertical
- horizontal

Description The set method null_format@ sets a chart's orientation, the direction of the chart's major axis.

See [ChartClass Methods](#) for more information.

-> proportional@

Sets proportional state

Class ChartClass set

Format this.proportional@(flag)

Arguments flag A Boolean value, TRUE means make the chart proportional when scaling.

Description The set method proportional@ sets the proportional state used when scaling the chart in the chart area.

See [ChartClass Methods](#) for more information.

-> scale_mode@

Sets scaling mode

Class ChartClass set

Format this.scale_mode@(mode)

Arguments mode The scaling mode, the valid modes are:

clip to fit Draw chart at full size, cutting off parts that do not fit the chart area

scale to fit Scale the chart to fit the chart area

scale if clips Scale the chart to fit the chart area if necessary, otherwise draw at full size

Description The set method scale_mode@ sets the mode used to scale the chart in the chart area.

See [ChartClass Methods](#) for more information.

-> titles@

Sets all of a chart's title information

Class ChartClass set

Format this.titles@(format chart_titles@ info)

Arguments info The title information.

Description The set method titles@ sets the text and alignment for a chart's title, subtitle, and footer, in chart_legend@ format. The chart_legend@ format is defined in the /install_dir/axdata/elf/chart_.am file. Include this file in any sources using the format. The format is defined as:

format chart_title@
title,

```

subtitle,
footer,
title_alignment,
    0    left
    1    center
    2    right
subtitle_alignment,
    0    left
    1    center
    2    right
footer_alignment
    0    left
    1    center
    2    right

```

See [ChartClass Methods](#) for more information.

-> title_component@

Sets title component information

Class ChartClass set

Format this.title_component@(item, string, align, mFlag, xOff, yOff)

Arguments

item	One of the following title components: title subtitle footer
string	The item string, or an array of 5 values. If an array is passed, the array contains the following values: string[0] Item string string[1] Item alignment. string[2] A Boolean value, TRUE means expand margin for item text. string[3] Item x-axis offset. string[4] Item y-axis offset.
align	Item alignment, one of the following values: 0 Left 1 Center

	2	Right
mFlag	A Boolean value, TRUE means expand margin for item text.	
xOff	Item x-axis offset.	
yOff	Item y-axis offset.	

Description The set method `title_component@` sets the title component information.
See [ChartClass Methods](#) for more information.

-> type@

Sets a chart's type

Class ChartClass set

Format `this.type@(formatString)`

Arguments `formatString` One of the chart types defined in `bldchrt_.am`.

Description The set method `type@` sets a chart's type. See [Chart Types](#) for a list of defined chart types.

See [ChartClass Methods](#) for more information.

-> zoom@

Sets the chart zoom

Class ChartClass set

Format `this.zoom@(val)`

Arguments `val` Numeric value indicating percentage of zoom.

Description The set method `zoom@` sets a chart's zoom in the display area..

See [ChartClass Methods](#) for more information.

BaseClass Methods

Methods in Applixware Builder usually consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

The BaseClass contains methods used by all objects in Applixware Builder. Use these methods from any of the Builder classes to perform basic actions.

BaseClass set (->) methods

[add_child@](#)

[application_name@](#)

[comments@](#)

[delete_child@](#)

[delete_children@](#)

[inherit@](#)

[load_methods_file@](#)

[name@](#)

[parent@](#)

[public_objvars@](#)

[quick_help@](#)

[timer@](#)

BaseClass get (<-) methods

[all_children@](#)

[application@](#)

[application_name@](#)

child@

children@

comments@

inherit@

is_child@

load methods file@

name@

parent@

public objvars@

quick help@

resource content@

sibling@

source_strings@

timer@

BaseClass events

error event

initialize event

terminate event

time out event

ApplicationClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use the ApplicationClass methods to control the actions and contents of an application. The ApplicationClass object is the top-level object in an application. An application can contain only one ApplicationClass object.

See [Applixware Builder ELF Macros](#) for application-related ELF macros.

ApplicationClass set (->) methods

[arg_var@](#)

[dbox_close@](#)

[dbox_disable@](#)

[dbox_enable@](#)

[dbox_open@](#)

[dbox_update@](#)

[default_error_handler@](#)

[is_windowless@](#)

[object_server_close@](#)

[object_server_open@](#)

[quit@](#)

[remote_object_destroy@](#)

[return_value@](#)

[send_message@](#)

[task_id@](#)

ApplicationClass get (<-) methods

[all_data_sets@](#)

[all_dboxes@](#)

all_printers@

all_rt_gateways@

arg_count@

arg_var@

arg_vars@

data set find@

data source find@

dbox find@

dbox is open@

dbox main@

dbox open@

error object@

remote object create@

remote object find@

rt_gateway find@

send message get response@

task id@

ApplicationClass events

error event

initialize event

message event

message respond event

terminate event

time out event

DatasetClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

Use DatasetClass methods to perform database operations. You can use the methods to enhance a data set created with the Source tool, or to programmatically retrieve database information.

DatasetClass set (->) methods

[auto_query@](#)

[binary_size_limit@](#)

[bootstrap@](#)

[conditions@](#)

[conditions_dlg@](#)

[connect@](#)

[current_record_value@](#)

[cursor_props@](#)

[datatype@](#)

[dates_as_dates@](#)

[date_format@](#)

[decrement_cursor@](#)

[decrement_position@](#)

[define_database@](#)

[disconnect@](#)

[displayed_columns@](#)

[display_binary_data@](#)

[edit apply only@](#)

[edit commit@](#)

[edit commit only@](#)

[edit delete@](#)

[edit insert@](#)

[edit rollback@](#)

[edit toggle enabled@](#)

[edit undo@](#)

[exec direct@](#)

[expressions@](#)

[fetch all records@](#)

[fetch size@](#)

[group by@](#)

[group by info@](#)

[having@](#)

[having info@](#)

[increment cursor@](#)

[increment position@](#)

[is editable by default@](#)

[is editing enabled@](#)

[is transactionless@](#)

[join@](#)

[join info@](#)

[load query from file@](#)

[lock_on_edit@](#)

[login@](#)

[login_dlg@](#)

[login_global@](#)

[login_info@](#)

[max_records@](#)

[no_duplicates@](#)

[numbers_as_numbers@](#)

[order_by@](#)

[position@](#)

[query_changed@](#)

[requery@](#)

[re_query_database@](#)

[set_cursor@](#)

[set_position@](#)

[sort_dlg@](#)

[sort_info@](#)

[source@](#)

[status_message@](#)

[tables@](#)

[temp_status_message@](#)

[timeout@](#)

[transpose_data@](#)

[trim_strings@](#)

value@

DatasetClass get (<-) methods

all values@

auto query@

binary size limit@

columns in table@

command object@

conditions@

connection@

connection info@

connection object@

current record value@

cursor props@

datatype@

dates as dates@

date format@

displayed columns@

display binary data@

exec direct@

expressions@

fetch size@

get cursor@

get position@

group by@

[group by info@](#)
[has more records@](#)
[having@](#)
[having info@](#)
[is editable by default@](#)
[is edited@](#)
[is editing enabled@](#)
[is transactionless@](#)
[join@](#)
[join info@](#)
[lock on edit@](#)
[login@](#)
[login dlg@](#)
[login global@](#)
[login info@](#)
[max records@](#)
[no duplicates@](#)
[numbers as numbers@](#)
[order by@](#)
[passwd@](#)
[position@](#)
[record count@](#)
[sort info@](#)
[source@](#)

[sql_heading@](#)

[tables@](#)

[tables_used@](#)

[timeout@](#)

[transpose_data@](#)

[trim_strings@](#)

[username@](#)

[value@](#)

[vendor@](#)

DatasetClass events

[error event](#)

[format data for dbms](#)

[format data for user](#)

[initialize event](#)

[terminate event](#)

[time out event](#)

DialogBoxClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use the DialogBoxClass methods and events to control the appearance and attributes of a dialog box in the application.

DialogBoxClass set (->) methods

[accept_pokes@](#)

[add_child@](#)

[arg_var@](#)

[arg_vars@](#)

[close@](#)

[close with return@](#)

[customize menu bar dlg@](#)

[data set@](#)

[delete_child@](#)

[disable@](#)

[display@](#)

[enable@](#)

[execute and dismiss@](#)

[height@](#)

[help topic@](#)

[icon name@](#)

[icon title@](#)

[is cancel destroy@](#)

[is expand@](#)

[keyfocus widget@](#)

[load custom menu bar@](#)

[menu bar@](#)

[open@](#)

post_help@
return_value@
send_poke@
title@
toggle_view_expressline@
type@
update@
update_children@
update_expressline@
variable_size_info@
width@
x_pos@
y_pos@

DialogBoxClass get (<-) methods

accept_pokes@
arg_var@
data_set@
expressline_status@
height@
help_topic@
icon_name@
icon_title@
is_cancel_destroy@
is_expand@

[is_open@](#)

[keyfocus_widget@](#)

[menu_bar@](#)

[open@](#)

[return_value@](#)

[title@](#)

[type@](#)

[width@](#)

[x_pos@](#)

[y_pos@](#)

DialogBoxClass events

[destroy_event](#)

[initialize_event](#)

[resize_event](#)

[time_out_event](#)

ControlClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

The ControlClass contains the methods and events used by all dialog box control objects.

ControlClass set (->) methods

[data_set@](#)

data set field name@

data set field name list@

data set field value@

data source macro@

data source type@

data source value@

display@

display map macro@

display suspend@

is grayed@

is hidden@

rt field name@

rt field value@

rt gateway@

rt record name@

title@

validator macro@

x_pos@

y_pos@

ControlClass get (<-) methods

data set@

data set field name@

data set field name list@

data set field value@

[data_source@](#)

[data_source_is_connected@](#)

[data_source_macro@](#)

[data_source_type@](#)

[data_source_value@](#)

[display_map_macro@](#)

[display_suspend@](#)

[is_grayed@](#)

[is_hidden@](#)

[rt_field_name@](#)

[rt_field_value@](#)

[rt_gateway@](#)

[rt_record_name@](#)

[title@](#)

[validator_macro@](#)

[x_pos@](#)

[y_pos@](#)

ControlClass events

[error_event](#)

[initialize_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

ButtonClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use a ButtonClass object to perform an immediate action, such as opening or closing a dialog box.

ButtonClass set (->) methods

[control_color@](#)

[control_color_cmyk@](#)

[control_color_is_workspace@](#)

[control_color_name@](#)

[control_color_rgb@](#)

[display@](#)

[height@](#)

[is_default@](#)

[is_drop_shadowed@](#)

[thickness@](#)

[title_color@](#)

[title_color_cmyk@](#)

[title_color_name@](#)

[title_color_rgb@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title font italic@](#)

[title font name@](#)

[title font size@](#)

[title is shadowed@](#)

[type@](#)

[value@](#)

[width@](#)

ButtonClass get (<-) methods

[control color@](#)

[height@](#)

[is default@](#)

[is drop shadowed@](#)

[thickness@](#)

[title color@](#)

[title font attrs@](#)

[title font bold@](#)

[title font italic@](#)

[title font name@](#)

[title font size@](#)

[title is shadowed@](#)

[type@](#)

[value@](#)

[width@](#)

ButtonClass events

[clicked_event](#)

[error_event](#)

[initialize_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

IconClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use an IconClass object to place a decorative image in a dialog box

IconClass set (->) methods

[display@](#)

[value@](#)

IconClass get (<-) methods

[value@](#)

IconClass events

[error_event](#)

[initialize_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

PanelClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use a panel to separate groups of controls in the dialog box.

PanelClass set (->) methods

[control_color@](#)

[control_color_cmyk@](#)

[control_color_is_workspace@](#)

[control_color_name@](#)

[control_color_rgb@](#)

[height@](#)

[style@](#)

[thickness@](#)

[width@](#)

PanelClass get (<-) methods

[control_color@](#)

[height@](#)

[style@](#)

[thickness@](#)

[width@](#)

PanelClass events

[error event](#)

[initialize event](#)

[resize event](#)

[terminate event](#)

[time out event](#)

[update event](#)

LabelClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use a label to define controls and areas in a dialog box.

LabelClass set (->) methods

[title_color@](#)

[title_color_cmyk@](#)

[title_color_name@](#)

[title_color_rgb@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title_font_italic@](#)

[title_font_name@](#)

[title font size@](#)

[title is shadowed@](#)

[value@](#)

LabelClass get (<-) methods

[title color@](#)

[title font attrs@](#)

[title font bold@](#)

[title font italic@](#)

[title font name@](#)

[title font size@](#)

[title is shadowed@](#)

[value@](#)

LabelClass events

[error event](#)

[initialize event](#)

[resize event](#)

[terminate event](#)

[time out event](#)

[update event](#)

ToggleButtonClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A toggle button can either be turned on or turned off. Use a toggle button control when you have a single item that can either be chosen or not chosen.

ToggleButtonClass set (->) methods

control_color@

control_color_cmyk@

control_color_is_workspace@

control_color_name@

control_color_rgb@

is_drop_shadowed@

is_on@

is_three_state@

mux_toggle@

title_color@

title_color_cmyk@

title_color_name@

title_color_rgb@

title_font_attrs@

title_font_bold@

title_font_italic@

title_font_name@

title_font_size@

title_is_shadowed@

value@

ToggleButtonClass get (<-) methods

[control_color@](#)

[is_drop_shadowed@](#)

[is_on@](#)

[is_three_state@](#)

[title_color@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title_font_italic@](#)

[title_font_name@](#)

[title_font_size@](#)

[title_is_shadowed@](#)

[value@](#)

ToggleButtonClass events

[changed_event](#)

[error_event](#)

[initialize_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

EntryFieldClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

An entry field is a text entry area where the user can type information. Use an entry field to get string information for use in an application.

EntryFieldClass set (->) methods

[button3 menu info@](#)

[control_color@](#)

[control_color cmyk@](#)

[control_color is workspace@](#)

[control_color name@](#)

[control_color rgb@](#)

[cursor_pos@](#)

[is_numeric@](#)

[is_optional@](#)

[is_password@](#)

[is_selected@](#)

[is_trimmed@](#)

[max_chars length@](#)

[text_color@](#)

[text_color cmyk@](#)

[text_color name@](#)

[text_color rgb@](#)

[text_font attrs@](#)

[text font bold@](#)

[text font italic@](#)

[text font name@](#)

[text font size@](#)

[text is shadowed@](#)

[thickness@](#)

[title color@](#)

[title color cmyk@](#)

[title color name@](#)

[title color rgb@](#)

[title font attrs@](#)

[title font bold@](#)

[title font italic@](#)

[title font name@](#)

[title font size@](#)

[title is shadowed@](#)

[title position@](#)

[valid chars@](#)

[value@](#)

[width@](#)

EntryFieldClass get (<-) methods

[control_color@](#)

[cursor_pos@](#)

[is_numeric@](#)

is_optional@
is_password@
is_trimmed@
max_chars_length@
text_color@
text_font_attrs@
text_font_bold@
text_font_italic@
text_font_name@
text_font_size@
text_is_shadowed@
title_color@
title_font_attrs@
title_font_bold@
title_font_italic@
title_font_name@
title_font_size@
title_is_shadowed@
title_position@
valid_chars@
value@
width@

EntryFieldClass events

button3_menu_state_event

[changed event](#)

[error event](#)

[focus in event](#)

[focus out event](#)

[initialize event](#)

[resize event](#)

[terminate event](#)

[time out event](#)

[typing event](#)

[update event](#)

OptionMenuClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

An option menu offers a menu of options from which to choose, with only the current choice visible in the dialog box.

OptionMenuClass set (->) methods

[control color@](#)

[control color cmyk@](#)

[control color is workspace@](#)

[control color name@](#)

[control color rgb@](#)

is_drop_shadowed@

text_color@

text_color_cmyk@

text_color_name@

text_color_rgb@

text_font_attrs@

text_font_bold@

text_font_italic@

text_font_name@

text_font_size@

text_is_shadowed@

text_strings@

title_color@

title_color_cmyk@

title_color_name@

title_color_rgb@

title_font_attrs@

title_font_bold@

title_font_italic@

title_font_name@

title_font_size@

title_is_shadowed@

value@

value_index@

OptionMenuClass get (-) methods

[control_color@](#)

[is_drop_shadowed@](#)

[text_color@](#)

[text_font_attrs@](#)

[text_font_bold@](#)

[text_font_italic@](#)

[text_font_name@](#)

[text_font_size@](#)

[text_is_shadowed@](#)

[text_strings@](#)

[title_color@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title_font_italic@](#)

[title_font_name@](#)

[title_font_size@](#)

[title_is_shadowed@](#)

[value@](#)

[value_index@](#)

OptionMenuClass events

[changed_event](#)

[error_event](#)

[initialize_event](#)

[resize event](#)

[terminate event](#)

[time out event](#)

[update event](#)

EditBoxClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

An edit box displays a large amount of text that you can manipulate. You can add or remove information in an edit box. The text in an edit box automatically wraps, so the text will adjust to any edits. A vertical scrollbar allows you to move to different parts of the text in the edit box.

EditBoxClass set (->) methods

[button3 menu info@](#)

[control color@](#)

[control color cmyk@](#)

[control color is workspace@](#)

[control color name@](#)

[control color rgb@](#)

[copy to clipboard@](#)

[cut to clipboard@](#)

[display@](#)

[height@](#)

[is mono space@](#)

[is read only@](#)

[paste from clipboard@](#)

[selection@](#)

[text color@](#)

[text color cmyk@](#)

[text color name@](#)

[text color rgb@](#)

[text cursor@](#)

[text font attrs@](#)

[text font bold@](#)

[text font italic@](#)

[text font name@](#)

[text font size@](#)

[text is shadowed@](#)

[thickness@](#)

[value@](#)

[width@](#)

EditBoxClass get (<-) methods

[control color@](#)

[cursor line number@](#)

[height@](#)

[is mono space@](#)

is_read_only@
selected_word@
selection@
text_color@
text_font_attrs@
text_font_bold@
text_font_italic@
text_font_name@
text_font_size@
text_is_shadowed@
thickness@
value@
width@
word_at_cursor@

EditBoxClass events

button3 menu state event
changed event
error event
focus in event
focus out event
initialize event
resize event
terminate event
time out event

[typing_event](#)

[update_event](#)

ListBoxClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A list box is a group of items displayed inside a box. You can use vertical and horizontal scroll bars to scroll through entries if necessary. Some list boxes allow more than one item to be chosen at a time, while other list boxes allow only one item to be chosen at a time.

ListBoxClass set (->) methods

[button3_menu_info@](#)

[control_color@](#)

[control_color_cmyk@](#)

[control_color_is_workspace@](#)

[control_color_name@](#)

[control_color_rgb@](#)

[height@](#)

[hscroll_is_enabled@](#)

[hscroll_length@](#)

[hscroll_origin@](#)

[is_mono_space@](#)

[is_multi_select@](#)

is_read_only@

text_color@

text_color_cmyk@

text_color_name@

text_color_rgb@

text_font_attrs@

text_font_bold@

text_font_italic@

text_font_name@

text_font_size@

text_is_shadowed@

text_strings@

thickness@

value@

value_index@

vscroll_is_enabled@

vscroll_length@

vscroll_origin@

width@

ListBoxClass get (<-) methods

control_color@

height@

hscroll_is_enabled@

hscroll_length@

[hscroll_origin@](#)

[is_mono_space@](#)

[is_multi_select@](#)

[is_read_only@](#)

[text_color@](#)

[text_font_attrs@](#)

[text_font_bold@](#)

[text_font_italic@](#)

[text_font_name@](#)

[text_font_size@](#)

[text_is_shadowed@](#)

[text_strings@](#)

[thickness@](#)

[value@](#)

[value_index@](#)

[vscroll_is_enabled@](#)

[vscroll_length@](#)

[vscroll_origin@](#)

[width@](#)

ListBoxClass events

[button3_menu_state_event](#)

[changed_event](#)

[error_event](#)

[double_click_event](#)

[initialize event](#)

[multi select event](#)

[resize event](#)

[stroke select event](#)

[terminate event](#)

[time out event](#)

[update event](#)

RadioBoxClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A radio button group is a group of items from which only a single item can be chosen at a time. Normally, radio button groups are a collection of logically associated items.

RadioBoxClass set (->) methods

[control_color@](#)

[control_color cmyk@](#)

[control_color is workspace@](#)

[control_color name@](#)

[control_color rgb@](#)

[text_color@](#)

[text_color cmyk@](#)

[text_color name@](#)

[text_color_rgb@](#)

[text_font_attrs@](#)

[text_font_bold@](#)

[text_font_italic@](#)

[text_font_name@](#)

[text_font_size@](#)

[text_is_shadowed@](#)

[text_strings@](#)

[title_color@](#)

[title_color_cmyk@](#)

[title_color_name@](#)

[title_color_rgb@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title_font_italic@](#)

[title_font_name@](#)

[title_font_size@](#)

[title_is_shadowed@](#)

[value@](#)

[value_index@](#)

RadioButtonClass get (<-) methods

[text_color@](#)

[text_font_attrs@](#)

[text_font_bold@](#)

text font italic@

text font name@

text font size@

text is shadowed@

text strings@

title color@

title font attrs@

title font bold@

title font italic@

title font name@

title font size@

title is shadowed@

value@

value index@

RadioButtonClass events

changed event

error event

initialize event

resize event

terminate event

time out event

update event

TableClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A table displays information in columns and rows. A table can have horizontal and vertical scroll bars so that you can move through the table information.

TableClass set (->) methods

[button3 menu info@](#)

[cell editing is allowed@](#)

[column resizing is allowed@](#)

[control color@](#)

[control color cmyk@](#)

[control color is workspace@](#)

[control color name@](#)

[control color rgb@](#)

[corner pixmaps@](#)

[data set field name list@](#)

[display@](#)

[goto cell@](#)

[heading info@](#)

[height@](#)

[horizontal grid is hidden@](#)

hscroll is enabled@
hscroll length@
hscroll origin@
insert text@
is editable@
is multi select@
marker pixmaps@
marker strings@
marker width@
model is data request@
one row@
row markers is suppressed@
row numbers is enabled@
rt field list@
rt record list@
selections@
table data@
text color@
text color cmyk@
text color name@
text color rgb@
text font attrs@
text font bold@
text font italic@

[text font name@](#)

[text font size@](#)

[text is shadowed@](#)

[thickness@](#)

[title color@](#)

[title color cmyk@](#)

[title color name@](#)

[title color rgb@](#)

[title font attrs@](#)

[title font bold@](#)

[title font italic@](#)

[title font name@](#)

[title font size@](#)

[title is shadowed@](#)

[top row@](#)

[value@](#)

[vertical grid is hidden@](#)

[vscroll is enabled@](#)

[vscroll length@](#)

[vscroll origin@](#)

[width@](#)

TableClass get (<-) methods

[cell editing is allowed@](#)

[column resizing is allowed@](#)

control_color@
data set field name list@
display_lines@
heading_info@
height@
horizontal grid is hidden@
hscroll is enabled@
hscroll length@
hscroll origin@
is_editable@
is_multi_select@
marker width@
model is data request@
rows data@
row markers is suppressed@
row numbers is enabled@
rt field list@
rt record list@
selections@
table data@
text_color@
text font attrs@
text font bold@
text font italic@

[text font name@](#)
[text font size@](#)
[text is shadowed@](#)
[thickness@](#)
[title color@](#)
[title font attrs@](#)
[title font bold@](#)
[title font italic@](#)
[title font name@](#)
[title font size@](#)
[title is shadowed@](#)
[top row@](#)
[value@](#)
[vertical grid is hidden@](#)
[vscroll is enabled@](#)
[vscroll length@](#)
[vscroll origin@](#)
[width@](#)

TableClass events

[button3 menu state event](#)
[button press event](#)
[cell changed event](#)
[cell focus in event](#)
[cell focus out event](#)

[column resize event](#)

[double click event](#)

[error event](#)

[initialize event](#)

[request data event](#)

[resize event](#)

[selection changed event](#)

[terminate event](#)

[time out event](#)

[typing even](#)

[update event](#)

CrossTableClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A cross table summarizes the data from two columns in a data set and displays it in a grid format. You can choose a row, column, and type of value for a cross table. A value uses an aggregate function, such as count(*) or avg(), to summarize cross table information.

CrossTableClass set (->) methods

[cell value string@](#)

[column data set field name@](#)

[create totals column@](#)

[create_totals_row@](#)

[row_data_set_field_name@](#)

CrossTableClass get (<-) methods

[cell_value_string@](#)

[column_data_set_field_name@](#)

[create_totals_column@](#)

[create_totals_row@](#)

[is_column_name@](#)

[row_data_set_field_name@](#)

CrossTableClass events

[button3_menu_state_event](#)

[button_press_event](#)

[cell_changed_event](#)

[cell_focus_in_event](#)

[cell_focus_out_event](#)

[column_resize_event](#)

[double_click_event](#)

[error_event](#)

[initialize_event](#)

[request_data_event](#)

[resize_event](#)

[selection_changed_event](#)

[terminate_event](#)

[time_out_event](#)

[typing_event](#)

[update_event](#)

PenClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use PenClass methods in conjunction with CanvasClass methods to render text and images on a CanvasClass object. Refer to XWindows information on graphics context (gc) for an understanding of how to use a PenClass object.

PenClass set (->) methods

[arc_mode@](#)

[background_color@](#)

[background_color_cmyk@](#)

[background_color_name@](#)

[background_color_rgb@](#)

[cap_style@](#)

[clip_mask@](#)

[clip_x_origin@](#)

[clip_y_origin@](#)

[dashes@](#)

[dash_offset@](#)

[fill_rule@](#)

[fill_style@](#)

[font_attrs@](#)

[font_bold@](#)

[font_italic@](#)

[font_name@](#)

[font_shadow@](#)

[font_size@](#)

[foreground_color@](#)

[foreground_color_cmyk@](#)

[foreground_color_name@](#)

[foreground_color_rgb@](#)

[graphics_exposures@](#)

[join_style@](#)

[line_style@](#)

[line_width@](#)

[plane_mask@](#)

[stipple@](#)

[subwindow_mode@](#)

[tile@](#)

[ts_x_origin@](#)

ts_y_origin@

xfer_func@

PenClass get (<-) methods

arc_mode@

background_color@

cap_style@

clip_mask@

clip_x_origin@

clip_y_origin@

dashes@

dash_offset@

fill_rule@

fill_style@

font_attrs@

font_bold@

font_italic@

font_name@

font_shadow@

font_size@

foreground_color@

graphics_exposures@

join_style@

line_style@

line_width@

[plane_mask@](#)

[stipple@](#)

[subwindow_mode@](#)

[tile@](#)

[ts_x_origin@](#)

[ts_y_origin@](#)

[xfer_func@](#)

PenClass events

[error event](#)

[initialize event](#)

[terminate event](#)

[time out event](#)

CanvasClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use PenClass methods in conjunction with CanvasClass methods to render text and images on a CanvasClass object. Refer to XWindows information on graphics context (gc) for an understanding of how to use a PenClass object with a CanvasClass object.

CanvasClass set methods have no effect until the canvas is displayed. Call set methods in the update_event after the canvas is displayed.

CanvasClass set (->) methods

[button3 menu info@](#)

clear@

clear area@

control color@

control color cmyk@

control color is workspace@

control color name@

control color rgb@

create inset@

draw arc@

draw icon@

draw inset@

draw line@

draw line from xys@

draw polygon@

draw rectangle@

draw text@

fill arc@

fill polygon@

fill rectangle@

height@

hscroll is enabled@

hscroll length@

hscroll origin@

inset file@

[paint_height@](#)

[paint_width@](#)

[scroll_incr@](#)

[scroll_page_incr@](#)

[scroll_pos@](#)

[thickness@](#)

[vscroll_is_enabled@](#)

[vscroll_length@](#)

[vscroll_origin@](#)

[width@](#)

CanvasClass get (<-) methods

[control_color@](#)

[height@](#)

[hscroll_is_enabled@](#)

[hscroll_length@](#)

[hscroll_origin@](#)

[icon_size@](#)

[inset_file@](#)

[inset_type@](#)

[measure_inset_object@](#)

[scroll_pos@](#)

[text_font_baseline@](#)

[text_font_height@](#)

[text_width@](#)

thickness@

vscroll is enabled@

vscroll length@

vscroll origin@

width@

CanvasClass events

button2 double click event

button2 motion event

button2 press event

button2 release event

button3 double click event

button3 menu state event

button3 motion event

button3 press event

button3 release event

button press event

button release event

double click event

error event

expose event

initialize event

keyboard event

motion event

resize event

[scroll event](#)

[terminate event](#)

[time out event](#)

[update event](#)

RealtimeGatewayClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use RealtimeGatewayClass methods to define a Real Time gateway in an application.

RealtimeGatewayClass set (->) methods

[add_record@](#)

[add_template@](#)

[connect@](#)

[delete_record@](#)

[delete_template@](#)

[disconnect@](#)

[gateway@](#)

[hostname@](#)

[register_all@](#)

[register_record@](#)

[unregister_all@](#)

[unregister_record@](#)

RealtimeGatewayClass get (<-) methods

[add_record@](#)

[add_template@](#)

[gateway@](#)

[hostname@](#)

[is_connected@](#)

[record_children@](#)

[rt_data_current_field_value@](#)

[template_children@](#)

RealtimeGatewayClass events

[error event](#)

[initialize event](#)

[terminate event](#)

[time out event](#)

RealtimeRecordClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use RealtimeRecordClass methods to define the information records contained in a RealtimeGatewayClass object.

RealtimeRecordClass set (->) methods

[add_field_item@](#)

[delete_field_item@](#)
[field_converter@](#)
[field_label@](#)
[field_list@](#)
[field_names_list@](#)
[field_publish_converter@](#)
[publish@](#)
[publish_field@](#)
[publish_fields@](#)
[publish_fields_by_dex@](#)
[publish_field_by_dex@](#)
[publish_value@](#)
[publish_value_array@](#)
[publish_value_by_dex@](#)
[record_id@](#)
[record_label@](#)
[register@](#)
[rtsql_database@](#)
[rtsql_host@](#)
[rtsql_query@](#)
[rtsql_routing@](#)
[rtsql_server@](#)
[rtsql_trigger@](#)
[rtsql_vendor@](#)

service_name@

template_name@

unregister@

use_template@

RealtimeRecordClass get (<-) methods

convert_publish_value@

convert_publish_value_by_dex@

current_value@

current_value_array@

current_value_by_dex@

display_value@

display_value_array@

display_value_by_dex@

field_converter@

field_label@

field_list@

field_names_list@

field_publish_converter@

is_registered@

publish_status@

publish_status_array@

publish_status_by_dex@

publish_value@

publish_value_array@

[publish value by dex@](#)

[record id@](#)

[record label@](#)

[rtsql database@](#)

[rtsql host@](#)

[rtsql query@](#)

[rtsql routing@](#)

[rtsql server@](#)

[rtsql trigger@](#)

[rtsql vendor@](#)

[service name@](#)

[template name@](#)

RealtimeRecordClass events

[data update event](#)

[error event](#)

[initialize event](#)

[publish status event](#)

[terminate event](#)

[time out event](#)

RealtimeTemplateClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use RealtimeTemplateClass methods to define a set of common field information you want to retrieve across a group of records.

RealtimeTemplateClass set (->) methods

[add field item@](#)

[delete field item@](#)

[field converter@](#)

[field label@](#)

[field list@](#)

[field names list@](#)

[field publish converter@](#)

RealtimeTemplateClass get (<-) methods

[field converter@](#)

[field label@](#)

[field list@](#)

[field names list@](#)

[field publish converter@](#)

RealtimeTemplateClass events

[error event](#)

[initialize event](#)

[terminate event](#)

[time out event](#)

ScalerClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use a scale to gradually increment and decrement values. A scale consists of a slider and sliding area. A scale can be used alone, or with an entry area to provide exact feedback on values. A scale is only used horizontally.

ScalerClass set (->) methods

[control_color@](#)

[control_color_cmyk@](#)

[control_color_is_workspace@](#)

[control_color_name@](#)

[control_color_rgb@](#)

[height@](#)

[max_value@](#)

[min_value@](#)

[value@](#)

[width@](#)

ScalerClass get (<-) methods

[control_color@](#)

[height@](#)

[max_value@](#)

[min_value@](#)

[value@](#)

[width@](#)

ScalerClass events

[changed_event](#)

[error_event](#)

[initialize_event](#)

[motion_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

ErrorClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

The ErrorClass methods allow you to write error handlers. If you do not catch errors, the Applixware Builder run time environment handles the errors. Write an ApplicationClass error_event to handle the error object created by the application when an error is thrown. If an error_event does not exist, the error is posted in a message box, then the application is terminated.

ErrorClass get (<-) methods

[error_file@](#)

[error_function@](#)

[error_line@](#)

[error_number@](#)

[error_object@](#)

[error_prepend@](#)

[error_string@](#)

ErrorClass events

[error_event](#)

[initialize_event](#)

[terminate_event](#)

[time_out_event](#)

MenuBarClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

The MenuBarClass methods allow you to define or control menu bar actions in a dialog box.

MenuBarClass set (->) methods

[data@](#)

[dbox_close@](#)

[dbox_open@](#)

[menu_item_name@](#)

[menu_item_status@](#)

MenuBarClass get (<-) methods

[data@](#)

MenuBarClass events

[error event](#)

[initialize event](#)

[menu name event](#)

[menu pick event](#)

[menu state event](#)

[terminate event](#)

[time out event](#)

[update event](#)

RowColClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A RowCol widget is similar to an option menu. Use a RowCol widget to display either text or bitmaps in an expandable, multiple column menu. An option menu is limited to a single column of text.

RowColClass set (->) methods

[background_color@](#)

[background_color_cmyk@](#)

[background_color_name@](#)

[background_color_rgb@](#)

[contents_type_is_pixmaps@](#)

[contents type is strings@](#)
[control color@](#)
[control color cmyk@](#)
[control color is workspace@](#)
[control color name@](#)
[control color rgb@](#)
[is drop shadowed@](#)
[number of columns@](#)
[pixmap@](#)
[pixmap labels@](#)
[style column major@](#)
[style row major@](#)
[style single column@](#)
[style single row@](#)
[style square@](#)
[style type@](#)
[text color@](#)
[text color cmyk@](#)
[text color name@](#)
[text color rgb@](#)
[text font attrs@](#)
[text font bold@](#)
[text font italic@](#)
[text font name@](#)

text font size@

text is shadowed@

text strings@

title color@

title color cmyk@

title color name@

title color rgb@

title font attrs@

title font bold@

title font italic@

title font name@

title font size@

title is shadowed@

value@

value index@

RowColClass get (<-) methods

background color@

contents type is pixmaps@

contents type is strings@

control color@

is drop shadowed@

number of columns@

pixmaps@

pixmap labels@

[style_type@](#)

[text_color@](#)

[text_font_attrs@](#)

[text_font_bold@](#)

[text_font_italic@](#)

[text_font_name@](#)

[text_font_size@](#)

[text_is_shadowed@](#)

[text_strings@](#)

[title_color@](#)

[title_font_attrs@](#)

[title_font_bold@](#)

[title_font_italic@](#)

[title_font_name@](#)

[title_font_size@](#)

[title_is_shadowed@](#)

[value@](#)

[value_index@](#)

RowColClass events

[changed_event](#)

[error_event](#)

[initialize_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

SplitterClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

A splitter is a diamond-shaped marker. Use a splitter to resize controls in a dialog box. A splitter moves along a vertical area within a dialog box.

SplitterClass set (->) methods

[marker_width@](#)

[max_value@](#)

[value@](#)

SplitterClass get (<-) methods

[marker_width@](#)

[max_value@](#)

[value@](#)

SplitterClass events

[changed_event](#)

[error_event](#)

[initialize_event](#)

[motion_event](#)

[picked_event](#)

[released_event](#)

[resize_event](#)

[terminate_event](#)

[time_out_event](#)

[update_event](#)

PrinterClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use PrinterClass objects to print the information contained in an application to PCL5 or PostScript printers.

PrinterClass set (->) methods

[copies@](#)

[cursor@](#)

[defaults@](#)

[draw_arc@](#)

[draw_chart@](#)

[draw_icon@](#)

[draw_line@](#)

[draw_polygon@](#)

[draw_rectangle@](#)

[draw_text@](#)

end_page@

fill_arc@

fill_polygon@

fill_rectangle@

footer@

header@

is_print_banner@

is_print_collated@

is_print_in_background@

is_print_in_color@

new_page@

page_dimensions@

page_margins@

page_metrics@

page_orientation@

page_setup_dlg@

page_size@

print@

printer_name@

printer_type@

print_file_name@

print_setup_dlg@

reset@

start_job@

[start_page@](#)

PrinterClass get (<-) methods

[copies@](#)

[cursor@](#)

[end_page@](#)

[icon_size@](#)

[is_print_banner@](#)

[is_print_collated@](#)

[is_print_in_background@](#)

[is_print_in_color@](#)

[is_started@](#)

[page_dimensions@](#)

[page_height@](#)

[page_margins@](#)

[page_number@](#)

[page_orientation@](#)

[page_size@](#)

[page_width@](#)

[preview_page_number@](#)

[printer_name@](#)

[printer_type@](#)

[print_file_name@](#)

[start_page@](#)

[text_size@](#)

PrinterClass events

[error event](#)

[initialize event](#)

[new page event](#)

[terminate event](#)

[time out event](#)

CommonDlgClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use CommonDlgClass methods to access the common application functions and dialog boxes.

CommonDlgClass set (->) methods

[applixware@](#)

[new application@](#)

[rt disable live feed@](#)

[rt display status@](#)

[rt enable live feed@](#)

[rt stop restart gateway dlg@](#)

[run application dlg@](#)

[run macro@](#)

[run_macro_dlg@](#)

[send_mail_dlg@](#)

CommonDlgClass get (<-) methods

[bitmap_dlg@](#)

[color_dlg@](#)

[is_application_running@](#)

[new_application@](#)

[open_dlg@](#)

[pend_for_new_application@](#)

[print_dlg@](#)

[rt_live_feed_is_enabled@](#)

[save_dlg@](#)

[system_builder_dir@](#)

CommonDlgClass events

[error_event](#)

[initialize_event](#)

[terminate_event](#)

[time_out_event](#)

MailClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use MailClass methods to create and send e-mail from an application.

MailClass set (->) methods

[alternate_reply_recipient@](#)

[attachments@](#)

[bc_recips@](#)

[body@](#)

[cc_recips@](#)

[defaults@](#)

[is_certified@](#)

[is_interactive@](#)

[is_reply_requested@](#)

[is_urgent@](#)

[outbox_copy@](#)

[reply_text@](#)

[reset@](#)

[send_mail@](#)

[subject@](#)

[to_recips@](#)

MailClass get (<-) methods

[alternate_reply_recipient@](#)

[attachments@](#)

[bc_recips@](#)

[body@](#)

[cc_recips@](#)

[is_certified@](#)

[is_interactive@](#)

[is_reply_requested@](#)

[is_urgent@](#)

[outbox_copy@](#)

[reply_text@](#)

[subject@](#)

[to_recips@](#)

MailClass events

[error_event](#)

[initialize_event](#)

[terminate_event](#)

[time_out_event](#)

ChartClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use PenClass methods in conjunction with ChartClass methods to draw charts. Refer to XWindows information on graphics context (gc) for an understanding of how to use a PenClass object with a ChartClass object.

ChartClass set (->) methods

[attributes@](#)

[attribute back fill@](#)
[attribute character@](#)
[attribute field@](#)
[attribute line fill@](#)
[attribute line style@](#)
[attribute shadow@](#)
[axis@](#)
[axis auto tik flags@](#)
[axis auto value flags@](#)
[axis bar spacing@](#)
[axis label@](#)
[axis labels@](#)
[axis line@](#)
[axis position@](#)
[axis tik@](#)
[axis tik label attrs@](#)
[axis values@](#)
[chart 3d info@](#)
[clear mode@](#)
[copy to clipboard@](#)
[create@](#)
[data@](#)
[data point label info@](#)
[decorations@](#)

[destroy@](#)
[display_chart@](#)
[drawing_area@](#)
[edit 3d dlg@](#)
[edit axes dlg@](#)
[edit frame and grid dlg@](#)
[edit layout dlg@](#)
[edit legend dlg@](#)
[edit titles dlg@](#)
[effects 3d@](#)
[enable 3d@](#)
[group@](#)
[group_axes@](#)
[group_label@](#)
[group_legend@](#)
[is data series column ordered@](#)
[is display on update@](#)
[is first col for legends@](#)
[is first row for labels@](#)
[legend@](#)
[legend_attr@](#)
[legend_box@](#)
[legend_display@](#)
[legend_title@](#)

margins@

null format@

orientation@

proportional@

scale mode@

titles@

title component@

type@

zoom@

ChartClass get (<-) methods

attributes@

attribute back fill@

attribute character@

attribute field@

attribute line fill@

attribute line style@

attribute shadow@

axes names@

axis@

axis auto tik flags@

axis auto value flags@

axis bar spacing@

axis create@

axis destroy@

[axis_label@](#)
[axis_labels@](#)
[axis_line@](#)
[axis_position@](#)
[axis_tik@](#)
[axis_tik_label_info@](#)
[axis_values@](#)
[chart_3d_info@](#)
[clear_mode@](#)
[data@](#)
[data_point_label_info@](#)
[decorations@](#)
[effects_3d@](#)
[group@](#)
[group_axes@](#)
[group_create@](#)
[group_create_from_data@](#)
[group_destroy@](#)
[group_label@](#)
[group_legend@](#)
[group_names@](#)
[is_3d_on@](#)
[is_data_series_column_ordered@](#)
[is_display_on_update@](#)

is first col for legends@

is first row for labels@

legend@

legend attr@

legend box@

legend display@

legend title@

margins@

null format@

orientation@

proportional@

scale mode@

titles@

title component@

type@

ChartClass events

button2 double click event

button2 motion event

button2 press event

button2 release event

button3 double click event

button3 menu state event

button3 motion event

button3 press event

[button3 release event](#)

[button press event](#)

[button release event](#)

[double click event](#)

[error event](#)

[expose event](#)

[initialize event](#)

[keyboard event](#)

[motion event](#)

[resize event](#)

[scroll event](#)

[terminate event](#)

[time out event](#)

[update event](#)

ComboBoxClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

ComboBoxClass set (->) methods

[control_color@](#)

[control_color cmyk@](#)

[control_color is workspace@](#)

control_color_name@

control_color_rgb@

editable@

text_color@

text_color_cmyk@

text_color_name@

text_color_rgb@

text_font_attrs@

text_font_bold@

text_font_italic@

text_font_name@

text_font_size@

text_is_shadowed@

text_strings@

value@

value_index@

width@

ComboBoxClass get (<-) methods

control_color@

editable@

text_color@

text_font_attrs@

text_font_bold@

text_font_italic@

[text font name@](#)

[text font size@](#)

[text is shadowed@](#)

[text strings@](#)

[value@](#)

[value index@](#)

[width@](#)

ComboBoxClass events

[changed event](#)

[error event](#)

[initialize event](#)

[resize event](#)

[terminate event](#)

[time out event](#)

[update event](#)

HistoricalDataClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

HistoricalDataClass set (->) methods

[add field@](#)

[add widget@](#)

date_order@

end_date@

fields@

period@

query@

record@

service@

show field names@

start_date@

toss null points@

widgets@

HistoricalDataClass get (<-) methods

date_order@

end_date@

fields@

period@

query@

record@

service@

show field names@

start_date@

toss null points@

widgets@

SQLCommandClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

SQLCommandClass set (->) methods

[apply_edits@](#)

[close_cursor@](#)

[delete_row@](#)

[display_binary_data@](#)

[fetch_all_rows@](#)

[insert_row@](#)

[lock_row@](#)

[next_pos@](#)

[open_cursor@](#)

[pos@](#)

[prepare@](#)

[put_binary_data@](#)

[unedit_row@](#)

[unprepare@](#)

[update_column@](#)

SQLCommandClass get (<-) methods

[column@](#)

cursor_status@

description@

display_binary_data@

fetchsize@

get_binary_data@

is_all_rows_fetched@

is_column_updatable@

is_row_locked@

is_statement_editable@

next_pos@

odbc_fetch_rowset@

pos@

row@

rows_fetched@

row_status@

SQLConnectClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

SQLConnectClass set (->) methods

close_gateway@

commit@

[connect@](#)

[disconnect@](#)

[exec_direct@](#)

[open_gateway@](#)

[rollback@](#)

SQLConnectClass get (<-) methods

[command@](#)

[database@](#)

[exec_direct@](#)

[host@](#)

[is_transactionless@](#)

[server@](#)

[transaction_state@](#)

[vendor@](#)

TabControlClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

TabControlClass set (->) methods

[active_layer@](#)

[insert_control@](#)

[layernames@](#)

load_dbox@

TabControlClass get (<-) methods

active_layer@

containees@

layernames@

top_inset@

TabControlClass events

error_event

initialize_event

resize_event

tab_event

terminate_event

time_out_event

update_event

GraphicsClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use GraphicsClass methods to program the Applixware Graphics application. The methods are the equivalents of the GR_ ELF macros.

GraphicsClass set (->) methods

[abandon@](#)

[abandon edits@](#)

[abandon pixels@](#)

[add bitmap@](#)

[add curve@](#)

[add page@](#)

[add selected to layer@](#)

[align@](#)

[apply attr@](#)

[array dup@](#)

[auto grid@](#)

[backspace key@](#)

[back return key@](#)

[bold@](#)

[borders@](#)

[center@](#)

[change callback@](#)

[change part@](#)

[chart change chart type@](#)

[chart create@](#)

[chart destroy@](#)

[chart draw@](#)

[chart rename@](#)

chart restore host@

chart save@

chart set 3d@

chart set attr@

chart set axis@

chart set axis labels@

chart set axis line@

chart set datum@

chart set decorations@

chart set extent@

chart set group@

chart set legend@

chart set margin@

chart set null format@

chart set orientation@

chart set title@

chart set type@

combine@

coordinates@

copy@

create color@

create layer@

create part@

cut@

[default_layer@](#)

[delete@](#)

[delete_char_key@](#)

[delete_doc@](#)

[delete_fill@](#)

[delete_layer@](#)

[delete_line@](#)

[delete_page@](#)

[delete_prev_char_key@](#)

[delete_selected_part@](#)

[delete_to_eol@](#)

[delete_word@](#)

[del_cmap_entry@](#)

[destroy_callback@](#)

[disable_handles@](#)

[disable_print_layer@](#)

[display_color_palette@](#)

[display_tool_palette@](#)

[down_arrow_key@](#)

[draw_by_handle@](#)

[draw_proportional@](#)

[duplicate_selected@](#)

[edit_arc@](#)

[edit_color@](#)

edit handout master@

edit outline@

edit outline master@

edit pixels@

edit sides@

edit slide@

edit slide master@

edit speaker notes@

edit speaker notes master@

edit text@

enable handles@

enable print layer@

enter help mode@

escape key@

exit@

export cgm@

export faxi@

export faxm@

export gif@

export gr21@

export gr31@

export hpgl@

export im@

export image@

[export iris@](#)

[export mswinbm@](#)

[export pbm@](#)

[export pgm@](#)

[export ppm@](#)

[export rs@](#)

[export tiffi@](#)

[export tiffm@](#)

[export wmf@](#)

[export xbm@](#)

[export xwd@](#)

[extrude@](#)

[first_page@](#)

[first screen key@](#)

[fit page in window@](#)

[fit text in path@](#)

[fit text to path@](#)

[goto begin line@](#)

[goto end line@](#)

[grids@](#)

[grid centimeter@](#)

[grid one eighth inch@](#)

[grid one half inch@](#)

[grid one quarter inch@](#)

grid one sixteenth inch@

grid one tenth inch@

group@

guides@

hide layer@

hide selected@

hide unselected@

horiz guide@

import cgm@

import dxf@

import eps@

import fax@

import gem@

import gif@

import gp4@

import hpgl@

import ilbm@

import im@

import iris@

import macpaint@

import pcx@

import pgm@

import pict2@

import ppm@

[import raw@](#)

[import sun@](#)

[import tga@](#)

[import tif@](#)

[import wmf@](#)

[import wpg@](#)

[import xbm@](#)

[import xpm@](#)

[import xwd@](#)

[is windowless@](#)

[italics@](#)

[larger@](#)

[last_page@](#)

[last screen key@](#)

[left arrow key@](#)

[left screen key@](#)

[load colormap file@](#)

[load custom fills@](#)

[load file@](#)

[load parts file@](#)

[lock layer@](#)

[lock selected@](#)

[lock tool@](#)

[mail@](#)

map icon cursor@

menu bar id@

merge file@

mouse double down@

mouse down@

mouse move@

mouse up@

move back@

move backward@

move exact@

move forward@

move front@

next page@

next page key@

next screen key@

pagebreaks@

paste@

paste eps@

paste fax@

paste gfx@

paste gr@

pick up attr@

previous page@

prev page key@

[prev_screen_key@](#)
[print@](#)
[px_bg_im_pel@](#)
[px_colorshift@](#)
[px_color_erase@](#)
[px_fg_im_pel@](#)
[px_find_edges@](#)
[px_flood@](#)
[px_invert@](#)
[px_restart@](#)
[px_set_brush_bg@](#)
[px_set_brush_fg@](#)
[px_set_flood_style@](#)
[px_set_pen_density@](#)
[px_set_pen_style@](#)
[px_set_tool@](#)
[px_shift_rgb@](#)
[px_soften@](#)
[px_toggle@](#)
[record_macro@](#)
[rename@](#)
[rename_fill@](#)
[render_graphic@](#)
[reopen@](#)

repaint window@

return key@

reveal@

reveal layer@

revert@

right arrow key@

right screen key@

rotate@

rotate exact@

rulers@

save@

save as@

scale exact@

select all@

select all in layer@

select clear@

select last@

select next@

select previous@

select reveal@

select reverse@

select text range@

select type@

set attr@

[set bg pixel color@](#)
[set callback@](#)
[set char atts@](#)
[set colormap@](#)
[set color by cmap index@](#)
[set display resolution@](#)
[set edit mode@](#)
[set ell@](#)
[set enable add mode@](#)
[set enable delete mode@](#)
[set enable select mode@](#)
[set face@](#)
[set fg pixel color@](#)
[set field atts@](#)
[set fill@](#)
[set fill bitmap@](#)
[set fill gradient@](#)
[set gradient@](#)
[set grid factor@](#)
[set headers and footers@](#)
[set hook@](#)
[set line atts@](#)
[set line bitmap@](#)
[set line fill@](#)

set line gradient@
set line width@
set page@
set page setup@
set palette line@
set percent fill@
set percent line fill@
set pixel color@
set point size@
set print info@
set ribbon info@
set rpoly@
set rrect@
set shadow@
set slide bg@
set slide color scheme@
set slide layout@
set slide transition@
set structured comment@
set tag@
set template bg@
set zoom@
shear@
slow motion@

[smaller@](#)

[smooth@](#)

[sort_parts@](#)

[sort_slides@](#)

[sticky_points@](#)

[suppress_drawing@](#)

[tab_key@](#)

[text_10_point@](#)

[text_12_point@](#)

[text_14_point@](#)

[text_18_point@](#)

[text_24_point@](#)

[text_36_point@](#)

[text_6_point@](#)

[text_8_point@](#)

[text_attrs@](#)

[text_avant_garde@](#)

[text_bookman@](#)

[text_chancery@](#)

[text_courier@](#)

[text_dingbats@](#)

[text_helvetica@](#)

[text_helv_narrow@](#)

[text_monospace@](#)

text palatino@
text sans serif@
text schoolbook@
text serif@
text special@
text symbol@
text times@
toggle select by handle@
tool lock@
tool pick@
type@
uncombine@
underline@
undo@
ungroup@
unlock all@
unlock layer@
unlock selected@
up arrow key@
vertical guide@
view expressline@
word back key@
word forward key@
x mirror@

[y_mirror@](#)

GraphicsClass get (<-) methods

[attach to task@](#)

[chart create axis@](#)

[chart create group@](#)

[chart destroy axis@](#)

[chart destroy group@](#)

[chart format number@](#)

[chart get 3d@](#)

[chart get attr@](#)

[chart get axes@](#)

[chart get axis@](#)

[chart get axis labels@](#)

[chart get axis line@](#)

[chart get charts@](#)

[chart get datum@](#)

[chart get decorations@](#)

[chart get extent@](#)

[chart get group@](#)

[chart get groups@](#)

[chart get legend@](#)

[chart get margin@](#)

[chart get null format@](#)

[chart get orientation@](#)

chart get title@
chart get type info@
chart get type name@
create callback@
current doc ptr string@
get area by handle@
get attr@
get backfill@
get bounder@
get callback@
get callbacks@
get char atts@
get color info@
get current chart@
get doc info@
get dpi@
get ell@
get field atts@
get fill name@
get fill style@
get font@
get grid factor@
get guides@
get headers and footers@

[get hook@](#)
[get layer info@](#)
[get linefill@](#)
[get marked position@](#)
[get mouse position@](#)
[get newest part@](#)
[get n colors in cmap@](#)
[get n fills@](#)
[get n layers@](#)
[get n pages@](#)
[get n pal parts@](#)
[get page@](#)
[get page setup@](#)
[get palette line@](#)
[get pal part name@](#)
[get rpoly@](#)
[get rrect@](#)
[get session host@](#)
[get shadow@](#)
[get slide bg@](#)
[get slide color scheme@](#)
[get slide layout@](#)
[get slide transition@](#)
[get structured comment@](#)

get_structured_comments@

get_tag@

get_template_bg@

get_tool_info@

get_udo_info@

list_font_families@

merge_attrs@

merge_lists@

mix_attrs@

modified@

olap_attrs@

pixel_edit_mode@

px_get_info@

px_modified@

query@

set_dpi@

GraphicsClass events

error_event

document_exit_event

document_modified_event

document_open_event

initialize_event

task_exit_event

terminate_event

[time_out_event](#)

SpreadsheetsClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use SpreadsheetsClass methods to program the Applixware Spreadsheets application. The methods are the equivalents of the SS_ ELF macros.

SpreadsheetsClass set (->) methods

[add_win@](#)

[backspace_key@](#)

[back_return_key@](#)

[blank@](#)

[blank_range@](#)

[bold@](#)

[bottom_section@](#)

[calc@](#)

[cell_file@](#)

[cell_justify@](#)

[change_named_range@](#)

[chart_change_chart_type@](#)

[chart_create@](#)

[chart_define_axis_range@](#)

[chart define axis title@](#)

[chart define data group@](#)

[chart define footer@](#)

[chart define legend labels@](#)

[chart define legend title@](#)

[chart define subtitle@](#)

[chart define title@](#)

[chart destroy@](#)

[chart draw@](#)

[chart format@](#)

[chart select@](#)

[chart set 3d@](#)

[chart set attr@](#)

[chart set axis labels@](#)

[chart set axis line@](#)

[chart set decorations@](#)

[chart set group@](#)

[chart set legend@](#)

[chart set margin@](#)

[chart set null format@](#)

[chart set orientation@](#)

[chart set title@](#)

[clear@](#)

[clear pagebreaks@](#)

[clear_protection@](#)
[close_objects@](#)
[copy@](#)
[create_chart@](#)
[create_named_range@](#)
[cut@](#)
[db_create_view@](#)
[db_delete_rec@](#)
[db_extract@](#)
[delete@](#)
[delete_cols@](#)
[delete_db@](#)
[delete_doc@](#)
[delete_key@](#)
[delete_line@](#)
[delete_object@](#)
[delete_range@](#)
[delete_rows@](#)
[delete_sheets@](#)
[delete_to_eol@](#)
[delete_word@](#)
[del_view@](#)
[down_arrow_key@](#)
[drag_inset_object@](#)

draw chart@

edit mode@

edit size profile@

enter key@

escape key@

exit@

fill@

find graphic object@

find replace@

goal seek@

goto begin line@

goto eol@

go begin@

go cell@

go end@

go name range@

go next selection@

go previous selection@

grid lines@

grid lines off@

grid lines on@

help key@

help mode@

hide@

[import_ascii@](#)

[import_col@](#)

[import_csv@](#)

[import_dif@](#)

[import_grid@](#)

[import_row@](#)

[import_sylk@](#)

[import_wk3@](#)

[import_wks@](#)

[import_xls@](#)

[insert@](#)

[insert_clipart@](#)

[insert_cols@](#)

[insert_line_break@](#)

[insert_object_at_cell@](#)

[insert_rows@](#)

[install_addin_functions@](#)

[invisible@](#)

[invisible_selected@](#)

[is_windowless@](#)

[italics@](#)

[justify_center@](#)

[justify_default@](#)

[justify_left@](#)

justify_repeat@

justify_right@

justify_selected@

larger_key@

left_arrow_key@

left_screen_key@

left_section@

load_file@

load_view@

localize_external_links@

localize_links@

localize_object_links@

mail@

menu_bar_id@

move_cells_down@

move_cells_left@

move_cells_right@

move_cells_up@

name_change@

name_change_ext@

name_create@

name_delete@

name_external@

new_sheet@

[next screen key@](#)
[number style@](#)
[num style@](#)
[object set title@](#)
[obj move back@](#)
[obj move backward@](#)
[obj move forward@](#)
[obj move front@](#)
[open object@](#)
[paste@](#)
[paste import@](#)
[paste import curdoc@](#)
[paste special@](#)
[preview@](#)
[preview close@](#)
[preview margins@](#)
[preview next page@](#)
[preview next view@](#)
[preview prev page@](#)
[preview prev view@](#)
[preview reformat@](#)
[preview repaint@](#)
[preview true zoom@](#)
[prev screen key@](#)

print@

print doc formulas@

proj_table1@

proj_table2@

protect@

put array formula@

put cell@

put range@

recalc@

record macro@

redisplay@

references cell@

register_function@

rename chart@

rename object@

repaint window@

return_key@

reveal@

reveal_some@

revert@

right_arrow_key@

right_screen_key@

right_section@

rt allow live toggle@

[rt live disable@](#)

[rt live enable@](#)

[save@](#)

[save as@](#)

[save profile@](#)

[save view@](#)

[search@](#)

[search next@](#)

[search next special@](#)

[search prev@](#)

[search prev special@](#)

[search special@](#)

[select all@](#)

[select all sheets@](#)

[select array formula@](#)

[select bottom section@](#)

[select clear@](#)

[select down@](#)

[select left@](#)

[select left section@](#)

[select material@](#)

[select object@](#)

[select pagebreaks@](#)

[select protected@](#)

[select right@](#)
[select right section@](#)
[select top section@](#)
[select up@](#)
[setcr value@](#)
[setptr value@](#)
[set ascii formatted@](#)
[set ascii quote labels@](#)
[set ascii trim whitespace@](#)
[set autobackup profile@](#)
[set auto resize rows@](#)
[set borders@](#)
[set calc options@](#)
[set callback@](#)
[set cell font size@](#)
[set chart group profile@](#)
[set chart navigate@](#)
[set chart navigate profile@](#)
[set col width@](#)
[set copied attrs profile@](#)
[set cr callback@](#)
[set currency@](#)
[set currency pos profile@](#)
[set currency symb profile@](#)

set cursor_profile@

set database@

set dbl click callback@

set delimiter_profile@

set doc_attr@

set drag_copypaste@

set drag_copypaste_profile@

set edit_face_profile@

set edit_size_profile@

set grid_lines@

set grid_profile@

set hdrftr@

set hook@

set initmacro_profile@

set linkupdate_profile@

set load_cell_callback@

set move_preference@

set move_preference_prof@

set object_hidden@

set object_locked@

set object_print@

set obj_info@

set open_cell_for_rng@

set open_cell_for_rng_prof@

set_pagebreaks@
set_page_setup@
set_protection@
set_pt_size@
set_range_attr@
set_range_pt_size@
set_record_mode@
set_record_mode_profile@
set_rowcol_headings@
set_rows_resize_profile@
set_row_height@
set_sheet@
set_sheets_display@
set_titles@
set_width@
set_worksheet_name@
set_zeroval_profile@
set_zero_vals@
set_zoom_factor@
show_pagebreaks@
smaller_key@
sort_cols@
sort_range@
sort_rows@

[special_enter_key@](#)

[tab_key@](#)

[text_10pt@](#)

[text_12pt@](#)

[text_14pt@](#)

[text_18pt@](#)

[text_24pt@](#)

[text_36pt@](#)

[text_6pt@](#)

[text_8pt@](#)

[text_avant_garde@](#)

[text_bookman@](#)

[text_chancery@](#)

[text_courier@](#)

[text_default@](#)

[text_dingbats@](#)

[text_gothic@](#)

[text_helvetica@](#)

[text_helv_narrow@](#)

[text_minchou@](#)

[text_monospace@](#)

[text_palatino@](#)

[text_sans_serif@](#)

[text_schoolbook@](#)

text_serif@
text_symbol@
text_times@
toggle_sync_scroll@
top_section@
type@
underline@
undo@
uninstall_addin_functions@
unregister_functions@
unset_callback@
unset_cr_callback@
unset_dbl_click_callback@
unset_load_cell_callback@
update_charts@
update_refs@
up_arrow_key@
view_expressline@
view_hide@
view_load_range@
view_range@
view_selected@
view_unhide@
visibility@

[visible@](#)

[visible selected@](#)

[wrap text@](#)

[zoom 100@](#)

[zoom 120@](#)

[zoom 150@](#)

[zoom 200@](#)

[zoom 300@](#)

[zoom 400@](#)

[zoom 40@](#)

[zoom 60@](#)

[zoom 80@](#)

SpreadsheetsClass get (<-) methods

[attach to task@](#)

[col num@](#)

[col string@](#)

[coordinate@](#)

[date string@](#)

[date value@](#)

[extract col row@](#)

[extract range info@](#)

[get borders@](#)

[get calc options@](#)

[get cell@](#)

get cell value@
get charts in doc@
get color table@
get col width@
get currency@
get db info@
get db names@
get doc attr@
get doc info@
get font families@
get font sizes@
get hdrftr@
get hook@
get links@
get named range info@
get names@
get names in doc@
get obj info@
get range prot@
get range prot vis@
get range style@
get row height@
get sheets@
get sheets display@

[get status@](#)

[get worksheet name@](#)

[get zoom factor@](#)

[insert sheets@](#)

[last cell info@](#)

[modified@](#)

[new win env@](#)

[object get names@](#)

[object get title@](#)

[open cell as string@](#)

[range@](#)

[rngstr to abs rngstr@](#)

[select@](#)

[sheet num@](#)

[sheet string@](#)

[view names@](#)

SpreadsheetsClass events

[error event](#)

[document exit event](#)

[document modified event](#)

[document open event](#)

[initialize event](#)

[task exit event](#)

[terminate event](#)

time_out_event

WordsClass Methods

Methods in Applixware Builder consist of set (->) and get (<-) methods.

- set methods set attributes and information for an object.
- get methods retrieve information from an object.

An event is a user-defined method. When an object action occurs, Applixware Builder calls the named event. Use methods and macros to define the actions the event performs.

Use WordsClass methods to program the Applixware Words application. The methods are the equivalents of the WP_ ELF macros.

WordsClass set (->) methods

access_user_dict@

add_color@

add_dict@

add_object@

add_tag@

adjust_cursor_for_inset@

align_center@

align_full@

align_left@

align_right@

apply_frame@

[apply_saved_atts@](#)

[apply_style@](#)

[backspace_key@](#)

[backward_search@](#)

[bold@](#)

[border@](#)

[border_bottom_style@](#)

[border_horizontal_margin@](#)

[border_horizontal_style@](#)

[border_left_style@](#)

[border_margins@](#)

[border_outline_style@](#)

[border_right_style@](#)

[border_shadow@](#)

[border_top_style@](#)

[border_vertical_margin@](#)

[border_vertical_style@](#)

[bullets@](#)

[calculate@](#)

[cell_list_box@](#)

[cell_manage_widget@](#)

[cell_radio_value@](#)

[cell_toggle@](#)

[center@](#)

[change bars@](#)

[change bars display@](#)

[change bars remove@](#)

[change bars write@](#)

[change link@](#)

[change link info@](#)

[clear field attr edits@](#)

[clear selection@](#)

[clear status line@](#)

[cmyk to rgb@](#)

[colon to table@](#)

[column break@](#)

[convert object@](#)

[copy@](#)

[copy colors@](#)

[copy glossaries@](#)

[copy row@](#)

[copy styles@](#)

[copy styles from file@](#)

[create glossary@](#)

[create glossary from beads@](#)

[create glossary from str@](#)

[create toc@](#)

[cut@](#)

[defer screen update@](#)

[define series@](#)

[delete@](#)

[delete bead@](#)

[delete beads@](#)

[delete cell@](#)

[delete cells@](#)

[delete char key@](#)

[delete color@](#)

[delete doc@](#)

[delete glossary@](#)

[delete keep@](#)

[delete line@](#)

[delete marker bead@](#)

[delete object@](#)

[delete row@](#)

[delete selected cells@](#)

[delete selection marks@](#)

[delete series@](#)

[delete style@](#)

[delete tabstop@](#)

[delete tag@](#)

[delete to end of line@](#)

[delete unused styles@](#)

delete word@

demote@

display location@

display loc of cursor@

display page of cursor@

doc is form@

down arrow key@

enlarge view scale@

enter char@

enter chars face@

enter help mode@

enter newline@

enter text@

escape key@

eval all fields@

eval fields@

exit@

export ascii layout@

export ascii lines@

export ascii paras@

export html@

export rtf@

export wp311@

find next@

[find_nth_cell@](#)
[find_pattern@](#)
[find_prev@](#)
[format_frame@](#)
[format_graphic@](#)
[format_section_type@](#)
[get_actual_frame_size@](#)
[get_bead_flow_bounds@](#)
[get_column_info@](#)
[get_current_column_info@](#)
[get_current_language@](#)
[get_doc_attrs@](#)
[get_doc_bounds@](#)
[get_flow_bounds@](#)
[get_hdrftr_margins@](#)
[get_lang_and_dicts@](#)
[get_last_row@](#)
[get_next_row@](#)
[get_page_numbering_format@](#)
[get_para_bullet@](#)
[get_para_numbering@](#)
[get_range_of_table@](#)
[goto_beginning@](#)
[goto_begin_line@](#)

goto bottom of page@

goto bottom of screen@

goto end@

goto end of line@

goto end of para@

goto even footer@

goto even header@

goto first cell@

goto first footer@

goto first header@

goto first page@

goto last footer@

goto last header@

goto last page@

goto next inset@

goto next line@

goto next page@

goto next para@

goto next screen@

goto next section@

goto next word@

goto nth para@

goto odd footer@

goto odd header@

[goto_page@](#)
[goto_page number@](#)
[goto_prev inset@](#)
[goto_prev line@](#)
[goto_prev_page@](#)
[goto_prev_para@](#)
[goto_prev_screen@](#)
[goto_prev_section@](#)
[goto_prev_word@](#)
[goto_start_of_line@](#)
[goto_start_of_para@](#)
[goto_top_of_page@](#)
[goto_top_of_screen@](#)
[go_para_num@](#)
[graphics_inset@](#)
[hard_space@](#)
[hsb_to_rgb@](#)
[import_ascii_lines@](#)
[import_asciiparas@](#)
[import_html@](#)
[import_rtf@](#)
[import_url@](#)
[import_winword@](#)
[import_wordperfect@](#)

indent@
inherit tabstops@
initialize doc@
insert advisory hyphen@
insert audio inset@
insert audio inset file@
insert cells at selection@
insert clipart@
insert cndI var field@
insert equation@
insert equation inset@
insert field date 0@
insert field date 10@
insert field date 11@
insert field date 1@
insert field date 2@
insert field date 3@
insert field date 4@
insert field date 5@
insert field date 6@
insert field date 7@
insert field date 8@
insert field date 9@
insert field page count@

[insert field page number@](#)

[insert field time 0@](#)

[insert footnote@](#)

[insert frame@](#)

[insert gr inset from gfx@](#)

[insert gr inset in frame@](#)

[insert line break@](#)

[insert link file field@](#)

[insert macro inset@](#)

[insert query inset@](#)

[insert section break@](#)

[insert ss inset@](#)

[insert table@](#)

[insert table cont text@](#)

[insert tabstop@](#)

[insert xrf src field@](#)

[install macros@](#)

[is windowless@](#)

[italics@](#)

[join@](#)

[justify center@](#)

[justify full@](#)

[justify left@](#)

[justify right@](#)

[left arrow key@](#)

[legalize location@](#)

[link_ext_ascii_grid@](#)

[link_ext_ascii_lines@](#)

[link_ext_ascii_paras@](#)

[link_ext_audio@](#)

[link_ext_axbitmap@](#)

[link_ext_cgm@](#)

[link_ext_csv@](#)

[link_ext_dif@](#)

[link_ext_dxf@](#)

[link_ext_eps@](#)

[link_ext_eqn@](#)

[link_ext_fax@](#)

[link_ext_gem@](#)

[link_ext_gif@](#)

[link_ext_gp4@](#)

[link_ext_gr@](#)

[link_ext_hpgl@](#)

[link_ext_html@](#)

[link_ext_ilbm@](#)

[link_ext_iris@](#)

[link_ext_mac_paint@](#)

[link_ext_me@](#)

[link_ext_misc@](#)

[link_ext_mswbmp@](#)

[link_ext_pbm@](#)

[link_ext_pcx@](#)

[link_ext_pgm@](#)

[link_ext_pict2@](#)

[link_ext_pict@](#)

[link_ext_ppm@](#)

[link_ext_query@](#)

[link_ext_raw@](#)

[link_ext_rtf@](#)

[link_ext_ss@](#)

[link_ext_ssgrid@](#)

[link_ext_sun_raster@](#)

[link_ext_sylk@](#)

[link_ext_tga@](#)

[link_ext_tiff@](#)

[link_ext_wfm5@](#)

[link_ext_wfw2@](#)

[link_ext_wk3@](#)

[link_ext_wk4@](#)

[link_ext_wks@](#)

[link_ext_wmf@](#)

[link_ext_wp@](#)

[link_ext_wpg@](#)

[link_ext_xbm@](#)

[link_ext_xls@](#)

[link_ext_xpm@](#)

[link_ext_xwd@](#)

[load_file@](#)

[localize_all_links@](#)

[localize_link@](#)

[mail@](#)

[make_passed_row_fit_std@](#)

[make_rows_fit@](#)

[make_row_fit_quick@](#)

[mark_point@](#)

[menu_bar_id@](#)

[menu_status_callback@](#)

[merge_cells@](#)

[merge_cells_first@](#)

[merge_cells_wide@](#)

[merge_init@](#)

[merge_row@](#)

[merge_selected_cells@](#)

[merge_selected_rows@](#)

[merge_terminate@](#)

[modify_border_thickness@](#)

[modify_color@](#)
[move_tabstop@](#)
[new@](#)
[next_key@](#)
[next_page@](#)
[numbering@](#)
[object allow unref@](#)
[object filter macro@](#)
[object launch macro@](#)
[object name@](#)
[object set temp file@](#)
[open@](#)
[outdent@](#)
[output beads@](#)
[overtyp toggle@](#)
[page number str of loc@](#)
[page setup@](#)
[paste@](#)
[paste cols after@](#)
[paste columns@](#)
[paste overwrite@](#)
[play audio@](#)
[preserve field attr edits@](#)
[prev key@](#)

prev_page@

print@

print device specific@

process bead actions@

process shift tab key@

process tab key@

promote@

protect current field@

protect field@

put range in row@

put text in cell@

quick bullets@

quick insert footer@

quick insert header@

quick numbering@

record macro@

redo@

reduce view scale@

remove row@

rename@

rename color@

repaint window@

replace bead@

replace selected@

[replace_style@](#)

[reset para settings@](#)

[reset text attrs@](#)

[restore selection@](#)

[restore tagged selection@](#)

[return key@](#)

[revert@](#)

[rgb to cmyk@](#)

[rgb to hsb@](#)

[right arrow key@](#)

[save@](#)

[save attrs@](#)

[save object@](#)

[save rename@](#)

[save version 3@](#)

[scroll to first page@](#)

[scroll to last page@](#)

[scroll to left screen@](#)

[scroll to next screen@](#)

[scroll to prev screen@](#)

[scroll to right screen@](#)

[selection larger@](#)

[selection smaller@](#)

[select all@](#)

select cell@
select column@
select current hdrftr@
select current para@
select field value@
select field value of field@
select left@
select line down@
select line up@
select para down@
select para up@
select prev section@
select range@
select range cells@
select range frame@
select right@
select row@
select table@
select to flow end@
select trailing spaces@
select whole lines@
select whole line down@
select whole line up@
select whole paras@

[select words@](#)

[series asterisks@](#)

[set all bounds display@](#)

[set ascii doc@](#)

[set breaks display@](#)

[set cell attrs@](#)

[set cell bounds display@](#)

[set cell margins@](#)

[set cell vert align@](#)

[set cell width@](#)

[set color table@](#)

[set column info@](#)

[set controls display@](#)

[set current border attrs@](#)

[set current column info@](#)

[set current gutter@](#)

[set current hdrftr info@](#)

[set current left margin@](#)

[set current page setup@](#)

[set current right margin@](#)

[set cursor@](#)

[set default pointer@](#)

[set dicts@](#)

[set doc attrs@](#)

set doc info@
set doc links info@
set doc var@
set doc window attr type@
set facing pages@
set fields display@
set flagged para attrs@
set flagged text attrs@
set form mode@
set frame bounds display@
set frame left edge@
set frame width@
set graphics display@
set hdrftr bounds display@
set hdrftr info@
set hdrftr margins@
set hook@
set inherits@
set language@
set links info@
set marker bead flag@
set modified@
set multiple selection@
set overwrite mode@

[set page bounds display@](#)
[set page numbering format@](#)
[set para bullet@](#)
[set para first indent@](#)
[set para left indent@](#)
[set para numbering@](#)
[set para right indent@](#)
[set para settings@](#)
[set print class@](#)
[set read only@](#)
[set row alignment@](#)
[set row attrs@](#)
[set row heading@](#)
[set row height@](#)
[set row height type@](#)
[set row indent@](#)
[set row margins@](#)
[set ruler decimal tab char@](#)
[set ruler display@](#)
[set section gutter@](#)
[set section left margin@](#)
[set section page setup@](#)
[set section right margin@](#)
[set selected cell attrs@](#)

set selected cell margins@
set selected cell v align@
set selected cell width@
set selected form attrs@
set selected row alignment@
set selected row attrs@
set selected row heading@
set selected row height@
set selected row ht type@
set selected row indent@
set selected row margins@
set selection marks@
set session var@
set skip dict@
set status line@
set status line temp@
set style@
set table cells width@
set table rows indent@
set tabstops@
set text attrs@
set text pointer@
set user dict@
set view char accurate@

[set view line accurate@](#)

[set view only@](#)

[set view state@](#)

[set window title@](#)

[shading@](#)

[shift case@](#)

[shift case in range@](#)

[shift cells@](#)

[shift down@](#)

[shift indents@](#)

[shift left@](#)

[shift name@](#)

[shift right@](#)

[shift selected@](#)

[shift up@](#)

[sort@](#)

[special chars@](#)

[spellcheck range@](#)

[spell word guess@](#)

[split bead@](#)

[split cell@](#)

[split cell at cursor@](#)

[split paragraph@](#)

[split row@](#)

split row after cell@

split row before cell@

split selected rows@

split selected table@

split table@

split table after row@

split table before row@

stamp new file@

swap beads@

swap cells@

swap tagged items@

table to text@

table to text quick@

tab key@

tag range@

temp save@

text 10 point@

text 12 point@

text 14 point@

text 18 point@

text 24 point@

text 36 point@

text 6 point@

text 8 point@

[text avant garde@](#)
[text bookman@](#)
[text chancery@](#)
[text courier@](#)
[text dict add word@](#)
[text dingbats@](#)
[text face@](#)
[text gothic@](#)
[text helvetica@](#)
[text helv narrow@](#)
[text minchou@](#)
[text palatino@](#)
[text schoolbook@](#)
[text size@](#)
[text symbol@](#)
[text times@](#)
[text to table@](#)
[text to table quick@](#)
[toggle all bounds display@](#)
[toggle breaks display@](#)
[toggle cell bounds display@](#)
[toggle controls display@](#)
[toggle fields display@](#)
[toggle form mode@](#)

toggle frame bounds display@

toggle graphics display@

toggle hf bounds display@

toggle page bounds display@

toggle ruler display@

to ascii@

transpose chars@

type@

underline@

undo@

undo activate@

undo clear@

undo end@

undo start@

unfield@

unique marker name@

up arrow key@

use style@

view 100@

view 120@

view 150@

view 200@

view 300@

view 400@

[view 40@](#)

[view 60@](#)

[view 80@](#)

[view char accurate@](#)

[view expressline@](#)

[view line accurate@](#)

[view scale@](#)

WordsClass get (<-) methods

[access special dict@](#)

[ascii template@](#)

[attach to task@](#)

[cell is selected@](#)

[cell widget value@](#)

[check doc legality@](#)

[col num@](#)

[compare index entries@](#)

[copy beads@](#)

[copy cell@](#)

[create button@](#)

[create index@](#)

[current page@](#)

[delete beads in range@](#)

[delete doc var@](#)

[delete session var@](#)

doc is open@
doc legality error@
does glossary exist@
edit object@
eqn_get current align@
eqn_get dflt size info@
eqn_get dflt spacing info@
eqn_get matrix info@
eqn_get size info@
eqn_get spacing info@
escape text@
extract index field items@
find bead class@
find bead of cell@
find bead of para@
find cell id@
find cell id in row@
find field of bead@
find first in para@
find flow bead@
find hyper target@
find last para@
find matching field end@
find matching field split@

[find_matching_field_start@](#)
[find_next_hyper_target@](#)
[find_next_para@](#)
[find_para_of_style@](#)
[find_prev_hyper_target@](#)
[find_prev_para@](#)
[find_style_bead@](#)
[find_visible_bead@](#)
[force_cursor@](#)
[format_number@](#)
[forward_search@](#)
[generate_new_object_name@](#)
[get_all_doc_var_names@](#)
[get_all_paras@](#)
[get_all_session_var_names@](#)
[get_all_tags@](#)
[get_ascii_doc@](#)
[get_available_width@](#)
[get_bead@](#)
[get_beads@](#)
[get_beads_in_range@](#)
[get_bead_class@](#)
[get_bead_classes@](#)
[get_bead_end@](#)

get cell@
get char at loc@
get color@
get color table@
get current border attrs@
get current hdrftr info@
get current page setup@
get current para range@
get current range@
get current section bead@
get current style name@
get current word@
get dicts@
get doc dict@
get doc links info@
get doc name for links@
get doc var@
get doc var names@
get doc window attr type@
get facing pages@
get field method@
get field method text@
get field nesting level@
get first row@

[get flagged para attrs@](#)

[get flagged text attrs@](#)

[get font families@](#)

[get form mode@](#)

[get frame attrs@](#)

[get glossary info@](#)

[get glossary refs in range@](#)

[get good language@](#)

[get graphic attrs@](#)

[get hdrftr from name@](#)

[get hdrftr info@](#)

[get hook@](#)

[get hyper target names@](#)

[get language@](#)

[get links@](#)

[get links info@](#)

[get marker bead@](#)

[get next bead character@](#)

[get next selection@](#)

[get nth para range@](#)

[get number format@](#)

[get number in range@](#)

[get number in text@](#)

[get num selections@](#)

get objects info@
get page of location@
get para settings@
get prev row@
get range of cell@
get range of cells@
get range of column@
get range of glossary@
get range of line@
get range of nth column@
get range of para@
get range of row@
get range of screen@
get range of word@
get read only@
get row@
get section page setup@
get selected cell attrs@
get selected form attrs@
get selected form info@
get selected row attrs@
get selected word@
get selection marks@
get session var@

[get session var names@](#)
[get skip dict@](#)
[get status@](#)
[get style@](#)
[get styles from file@](#)
[get style names@](#)
[get tag@](#)
[get tag bead@](#)
[get tag location@](#)
[get tag num@](#)
[get text attrs@](#)
[get text of cell@](#)
[get text of current para@](#)
[get text of range@](#)
[get typing text attrs@](#)
[get undo nesting@](#)
[get undo status@](#)
[get user dict@](#)
[get view only@](#)
[get view scale@](#)
[get view state@](#)
[get window height@](#)
[get window width@](#)
[get xref source names@](#)

goto_next_place@
goto_prev_place@
have_selection@
hyperlink@
illegalize_location@
illegal_for_table@
init_paragraph_bead@
init_para_attrs@
init_para_attr_flags@
init_style_bead@
init_text_attrs@
init_text_attr_flags@
init_text_bead@
insert_bead@
insert_beads@
insert_cells@
insert_column_break@
insert_field@
insert_gr_inset@
insert_marker_bead@
insert_page_break@
insert_page_break_next@
insert_rows@
insert_section@

[insert table at loc@](#)

[in footnote@](#)

[in hdrftr@](#)

[in main flow@](#)

[in selection marks@](#)

[is bead in cell@](#)

[is bead in table@](#)

[is child@](#)

[is child link@](#)

[is child object@](#)

[is eqn@](#)

[is face latin@](#)

[is help@](#)

[is me@](#)

[is wp@](#)

[list glossary@](#)

[list series@](#)

[local text attrs for array@](#)

[location in range@](#)

[location of cursor@](#)

[make row fit@](#)

[merge num records@](#)

[merge record@](#)

[merge record num@](#)

modified@
move beads@
new window@
next field gloss@
next field string@
next visible bead@
num cells in row@
num cols selected@
num rows in table@
num tags@
object is referenced@
open inset@
page number of loc@
preserve selection@
prev visible bead@
protect query@
range is cursor@
replace all@
resolve para tabs@
resolve style tabs@
retrieve tagged range@
row_num@
selection is multiple@
selection type for borders@

[select next cell@](#)

[select next section@](#)

[select prev cell@](#)

[series days@](#)

[series months@](#)

[set glossary info@](#)

[shift@](#)

[status line@](#)

[style effective atts@](#)

[style inherit atts@](#)

[tag selection@](#)

[undo status@](#)

[unescape text@](#)

[unfield query@](#)

[valid footnote location@](#)

[word count@](#)

WordsClass events

[error event](#)

[document exit event](#)

[document modified event](#)

[document open event](#)

[initialize event](#)

[task exit event](#)

[terminate event](#)

time_out_event

<- control_color@

Returns color used by object

Class ComboBoxClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the combo box. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [ComboBoxClass Methods](#) for more information.

<- editable@

Returns editable state

Class ComboBoxClass get

Format flag = this.editable@

Description The get method `editable@` returns a Boolean value indicating the editable state of the combo box. The method returns TRUE if the combo box is editable, FALSE otherwise.

See [ComboBoxClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class ComboBoxClass get

Format colors = this.text_color@

Description The get method `text_color@` returns a two dimensional array of text color settings. Text appears in the combo box.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [ComboBoxClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class ComboBoxClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the combo box. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name The font name.

font_size The font point size.

bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

shadow Not used.

See [ComboBoxClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class ComboBoxClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. You can use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the combo box.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [ComboBoxClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class ComboBoxClass get

Format flag = this.text_font_italic@

Description The get method text_font_italic@ returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method text_font_italic@ to verify and change the object text italic state.

Text appears in the combo box.

For example, to make the object text italic use the following:

```
var flag
```

```
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [ComboBoxClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class ComboBoxClass get

Format name = this.text_font_name@

Description The get method text_font_name@ returns the object text font name. You can use this method with the set method text_font_name@ to verify and change the object text font.

Text appears in the combo box.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"  'set method
```

See [ComboBoxClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class ComboBoxClass get

Format size = this.text_font_size@

Description The get method text_font_size@ returns the object text font size. You can use this method with the set method text_font_size@ to verify and change the object text size.

Text appears in the combo box.

For example, to set the object text size to 12 use the following:

```
var size
```

```
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [ComboBoxClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class ComboBoxClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the combo box. Text only appears shadowed on color displays.

See [ComboBoxClass Methods](#) for more information.

<- text_strings@

Returns combo box choices

Class ComboBoxClass get

Format valArray = this.text_strings@

Description The get method `text_strings@` returns the combo box choices as an array of strings. You can use this method with the set method `text_strings@` to verify and change the combo box choices.

For example, to set the combo box choices you would use the method as follows:

```
IF this.text_strings@ = NULL           'get method
    this.text_strings@ = {"red", "white", "blue"} 'set method
```

See [ComboBoxClass Methods](#) for more information.

<- value@

Returns current combo box choice

Class ComboBoxClass get

Format value = this.value@

Description The get method value@ returns the current combo box choice as a string. You can use this method with the set method value@ to verify and change the combo box choices.

For example, to set the current combo box choice you would use the method as follows:

```
IF this.value@ <> "blue"           'get method
    this.value@ = "blue"           'set method
```

See [ComboBoxClass Methods](#) for more information.

<- value_index@

Returns array index of combo box choice

Class ComboBoxClass get

Format index = this.value_index@

Description The get method value_index@ returns the array index of the current combo box choice as a numeric value. The combo box choices are a 0-based text string array.

If the combo box is editable, and a user-defined string is typed in the combo box that does not match one of the combo box text strings, this method returns -1. See the set method [editable@](#) for information about enabling combo box editing.

For example, to get the index of the current combo box choice you would use the method as follows:

```
var values, index
values = this.text_strings@
index = this.value_index@
if index <> -1{
    IF values[index] <> "blue"
        this.value@ = "blue"
```

```

    }
    else{ 'add the new string as a valid text string
          values[array_size@(values)] = this.value@
          this.text_strings@ = values
    }

```

See [ComboBoxClass Methods](#) for more information.

<- width@

Returns combo box width

Class ComboBoxClass get

Format pixels = this.width@

Description The get method width@ returns the combo box width in pixels. Use this method with the set method width@ to verify and change the combo box's width.

For example, to make the combo box width at least 250 pixels use the following:

```

var pixels
pixels = this.width@           'get method
IF pixels< 250
    this.width@ = 250         'set method

```

See [ComboBoxClass Methods](#) for more information.

-> control_color@

Sets control color

Class ComboBoxClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type

The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values.

See [ComboBoxClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class ComboBoxClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments cyan The CMYK cyan value.

magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [ComboBoxClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class ComboBoxClass set

Format `this.control_color_is_workspace@(flag)`

Arguments `flag` Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.

Description The set method `control_color_is_workspace@` sets the color used by the object. See [ComboBoxClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class ComboBoxClass set

Format `this.control_color_name@(name)`

Arguments `name` The color name.

Description The set method `control_color_name@` sets the control color by name. See [ComboBoxClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class ComboBoxClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the control color with RGB values.
See [ComboBoxClass Methods](#) for more information.

-> display@

Displays combo box

Class ComboBoxClass set

Format this.display@

Description The set method display@ displays the combo box in the dialog box.
See [ComboBoxClass Methods](#) for more information.

-> editable@

Sets combo box editable state

Class ComboBoxClass set

Format this.editable@(flag)

Arguments flag A Boolean value. TRUE makes the combo box editable.

Description The set method editable@ sets the combo box editable state. Use an editable combo box to accept user-defined defined input in your application, in addition to supplying default choices.

A user-defined string typed in an editable combo box that does not match one of the combo box text strings has a [value_index@](#) of -1.

See [ComboBoxClass Methods](#) for more information.

-> is_drop_shadowed@

Sets combo box shadowed state

Class ComboBoxClass set

Format this.is_drop_shadowed@(flag)

Arguments flag A Boolean value. TRUE makes the combo box drop shadowed.

Description The set method is_drop_shadowed@ sets the combo box shadowed state. A drop shadow gives a combo box a 3-dimensional appearance.

See [ComboBoxClass Methods](#) for more information.

-> text_color@

Sets text color

Class ComboBoxClass set

Format this.text_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.

- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears in the combo box.

See [ComboBoxClass Methods](#) for more information.

-> `text_color_cmyk@`

Sets text CMYK color

Class `ComboBoxClass` set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears in the combo box.

See [ComboBoxClass Methods](#) for more information.

-> `text_color_name@`

Sets text name color

Class `ComboBoxClass` set

Format `this.text_color_name@(name)`

Arguments

<code>name</code>	The color name.
-------------------	-----------------

Description The set method `text_color_name@` sets the text color by name. Text appears in the combo box.

See [ComboBoxClass Methods](#) for more information.

-> `text_color_rgb@`

Sets control RGB color

Class ComboBoxClass set

Format `this.text_color_rgb@(red, green, blue)`

Arguments

red	The RGB red value.
green	The RGB blue value.
blue	The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the combo box.

See [ComboBoxClass Methods](#) for more information.

-> `text_font_attrs@`

Sets text font information

Class ComboBoxClass set

Format `this.text_font_attrs@(format font_attrs_info@ fonts)`

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `text_font_attrs@` sets all the text font information. Text appears in the combo box. The `font_attrs_info@` format is defined in the `install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [ComboBoxClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class ComboBoxClass set

Format `this.text_font_bold@(flag)`

Arguments flag Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.

Description The set method `text_font_bold@` sets the text bold state. Text appears in the combo box. You can use this method with the get method `text_font_bold@` to verify and change the object text bold state. Use the set method `text_font_attrs@` to set all font information in one step.

See [ComboBoxClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class ComboBoxClass set

Format `this.text_font_italic@(flag)`

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method `text_font_italic@` sets the italic state of the object text. Text appears in the combo box. You can use this method with the get method `text_font_italic@` to verify and change the object text italic state. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
```

```
flag = this.text_font_italic@           'get method
```

IF NOT flag

```
this.text_font_italic@ = TRUE
```

'set method

See [ComboBoxClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class ComboBoxClass set

Format this.text_font_name@(name)

Arguments name A string for the font.

Description The set method text_font_name@ returns the object text font name. Text appears in the combo box. You can use this method with the get method text_font_name@ to verify and change the object text font. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
```

```
name = this.text_font_name@
```

'get method

```
IF name <> "Courier"
```

```
    this.text_font_name@ = "Courier" 'set method
```

See [ComboBoxClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class ComboBoxClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears in the combo box. You can use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [ComboBoxClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class ComboBoxClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method text_is_shadowed@ sets the shadow state of the object text. Use this method with the get method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the combo box. Text only appears shadowed on color displays.

See [ComboBoxClass Methods](#) for more information.

-> text_strings@

Sets combo box choices

Class ComboBoxClass set

Format this.text_strings@(valArray)

Arguments valArray An array of strings for the combo box choices.

Description The set method text_strings@ sets the combo box choices. You can use this method with the get method text_strings@ to verify and change the combo box choices.

For example, to set the combo box choices you would use the method as follows:

```
IF this.text_strings@ = NULL           'get method
```

```
this.text_strings@ = {"red", "white", "blue"}    'set method
```

See [ComboBoxClass Methods](#) for more information.

-> value@

Sets current combo box choice

Class ComboBoxClass set

Format this.value@(value)

Arguments value A string for the current combo box choice.

Description The set method value@ sets the current combo box choice.

For example, to set the current combo box choice you would use the method as follows:

```
IF this.value@ <> "blue"                            'get method
    this.value@ = "blue"                            'set method
```

See [ComboBoxClass Methods](#) for more information.

-> value_index@

Sets current combo box choice to array index

Class ComboBoxClass set

Format this.value_index@(index)

Arguments index A numeric value.

Description The set method value_index@ sets the current combo box choice to the array string index of the passed value. The combo box choices are a 0-based text string array.

For example, to set the index of the current combo box choice to the array index of the string blue you would use the method as follows:

```
var index, values
values = this.text_strings@
FOR index = 0 to ARRAY_SIZE@(values)-1
    IF values[index] = "blue"
        this.value_index@ = index
```

NEXT index

See [ComboBoxClass Methods](#) for more information.

-> width@

Sets combo box width

Class ComboBoxClass set

Format this.width@(pixels)

Arguments pixels The width, in pixels, of the combo box.

Description The set method width@ sets the combo box's width. Use this method with the get method width@ to verify and change the combo box's width.

For example, to make the combo box width at least 250 pixels use the following:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [ComboBoxClass Methods](#) for more information.

-> changed_event

Called when combo box choice changes

Class ComboBoxClass event

Format this.changed_event(index)

Arguments index A numeric value indicating the new menu choice index position. The combo box choices are a 0-based text string array.

Description The set event changed is called by Applixware Builder when the the combo box choice changes. This is a user-defined event, place all actions you want performed in the event definition. The following is an example of a combo box changed event.

```
set changed_event(index)
    IF value = "blue"
```

```

        {
            ' actions when combo box choice is blue
        }
    ELSE ' value is another combo box choice
    {
        ' actions when combo box choice is not blue
    }
endset

```

See [ComboBoxClass Methods](#) for more information.

<- error_event

Called for system errors

Class ComboBoxClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ComboBoxClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class ComboBoxClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the combo box color, position, and so on.

See [ComboBoxClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class ComboBoxClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The set event resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [ComboBoxClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class ComboBoxClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ComboBoxClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class ComboBoxClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [ComboBoxClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class ComboBoxClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [ComboBoxClass Methods](#) for more information.

<- bitmap_dlg@

Displays Icon Viewer dialog box

Class CommonDlgClass get

Format icon = this.bitmap_dlg@

Description The get method bitmap_dlg@ displays the Icon Viewer dialog box. If an icon is chosen with the dialog box, the icon name is returned, otherwise the method returns NULL.

See [CommonDlgClass Methods](#) for more information.

<- color_dlg@

Displays Choose Color dialog box

Class CommonDlgClass get

Format info = this.color_dlg@(args)

Arguments args An array of color information, in the following format:

args[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- args[1][0] The RGB red value.
- args[1][1] The RGB blue value.
- args[1][2] The RGB green value.

The following values apply to CMYK color type:

- args[1][0] The CMYK cyan value.
- args[1][1] The CMYK magenta value.
- args[1][2] The CMYK yellow value.
- args[1][3] The CMYK black value.

The following values apply to HSB color type:

- args[1][0] The HSB hue value.
- args[1][1] The HSB saturation value.
- args[1][2] The HSB brightness value.

For a named color type, `arg[1]` is a string for the color name.

Description The get method `color_dlg@` displays the Choose Color dialog box. If a color is chosen with the dialog box, the color values are returned, otherwise the method returns NULL.

See [CommonDlgClass Methods](#) for more information.

<- is_application_running@

Displays Icon Viewer dialog box

Class CommonDlgClass get

Format `flag = this.is_application_running@(object app)`

Arguments `app` An Applixware Builder application object.

Description The get method `is_application_running@` returns application status as a Boolean value. The method returns TRUE if the passed application object is running, otherwise the method returns FALSE.

See [CommonDlgClass Methods](#) for more information.

<- new_application@

Runs an application in a new task

Class CommonDlgClass get

Format `object app = this.new_application@(file[, arg1[, arg2[, arg3[, arg4[, arg5]]]])`

Arguments `file` The Applixware Builder file, with full file path. If file is passed as an array, `file[0]` is used as the file name, and all other array items are taken as application arguments.

`arg1` An optional application argument.

`arg2` An optional application argument.

`arg3` An optional application argument.

`arg4` An optional application argument.

`arg5` An optional application argument.

Description The get method `new_application@` runs an Applixware Builder application in a new task and returns the application object. Use the method to run multiple applications against

the same axmain process. The data types of the optional arguments are defined by the target application.

See [CommonDlgClass Methods](#) for more information.

<- open_dlg@

Displays Open dialog box

Class CommonDlgClass get

Format path= this.open_dlg@([suffix[, dir[, wildcard[, title[, helpid]]]])

Arguments

suffix	The file suffix, such as ".ab" for Applixware Builder files.
dir	The full path name of the directory.
wildcard	The search wildcard, such as *, ?, and so on.
title	The title to use in the Open dialog box. The title Open is used if no title is passed.
helpid	The help ID to pass to the Applixware Help system when the Help button is clicked in the dialog box.

Description The get method open_dlg@ displays the Open dialog box and returns the file name with full path name. This is the same dialog box used in Applixware Builder.

See [CommonDlgClass Methods](#) for more information.

<- pend_for_new_application@

Runs an application as a pended task

Class CommonDlgClass get

Format retVal = this.pend_for_new_application@(file[, arg1[, arg2[, arg3[, arg4[, arg5]]]])

Arguments

file	The Applixware Builder file, with full file path. If file is passed as an array, file[0] is used as the file name, and all other array items are taken as application arguments.
arg1	An optional application argument.
arg2	An optional application argument.
arg3	An optional application argument.

arg4 An optional application argument.
arg5 An optional application argument.

Description The get method `pend_for_new_application@` runs an Applixware Builder application as a pended task. The application from which you call the method waits for a return signal from the pended task. The data types of the optional arguments are defined by the target application. The data type of the return value is defined by the pended application.

See [CommonDlgClass Methods](#) for more information.

<- print_dlg@

Displays Print dialog box

Class CommonDlgClass get

Format info = this.print_dlg@(args)

Arguments args An array of arguments. The array contains the following arguments:

args[0]	Number of copies (integer)
args[1]	Start page (integer)
args[2]	Endpage (integer)
args[3]	Banner page (Boolean)
args[4]	Collate (Boolean)
args[5]	Color (Boolean)
args[6]	Background (Boolean)
args[7]	Printer type ("PostScript" or "PCL5")
args[8]	Printer name (string)
args[9]	Print file name (string)

Description The get method `print_dlg@` displays the Print dialog box. Click on Print in the dialog box to print the current material with the parameters reflected in the dialog box. Click on OK to set the print parameters without printing the material.

The method returns the settings in the dialog box as an array of arguments. The array contains additional arguments, `info[10]` is a Boolean value indicating if the file was sent to print. TRUE means the Print button was clicked, FALSE means the OK button was clicked.

info[11] is the printer information and options set in the dialog box, returned in `print_baggage@` format. The format is defined in the `install_dir/axdata/elf/print_.am` header file.

See [CommonDlgClass Methods](#) for more information.

`<- rt_live_feed_is_enabled@`

Returns Real Time server status

Class CommonDlgClass get

Format flag = this.rt_live_feed_is_enabled@

Description The get method `rt_live_feed_is_enabled@` returns the status of a Real Time server as a Boolean value. The method returns TRUE if the current task's interaction with the Real Time server is enabled, otherwise the method returns FALSE. Use the set methods [rt_disable_live_feed@](#) and [rt_enable_live_feed@](#) to set the task's interaction with the Real Time server.

See [CommonDlgClass Methods](#) for more information.

`<- save_dlg@`

Displays Save As dialog box

Class CommonDlgClass get

Format format doc_format info = this.save_dlg@([format doc_format docinfo[, suffix[, title[, app]]]])

Arguments

docinfo	An array of information in doc_format_ format. The format is defined in the ELF header file <code>fileinf_.am</code> , located in the <code>install_dir/axdata/elf</code> directory.
suffix	The file suffix to use as a search wildcard.
title	The title to use in the Save As dialog box. The title Save As is used if no title is passed.
app	An array of application specific information.

Description The get method `save_dlg@` displays the Save As dialog box and returns the file information in doc_format format. This is the same dialog box used in Applixware Builder.

See [CommonDlgClass Methods](#) for more information.

<- system_builder_dir@

Returns Applixware Builder directory

Class CommonDlgClass get

Format dir = this.system_builder_dir@

Description The get method system_builder_dir@ returns the directory where Applixware Builder is installed.

See [CommonDlgClass Methods](#) for more information.

-> Applixware@

Starts Applixware

Class CommonDlgClass set

Format this.Applixware@

Description The set method Applixware@ starts an Applixware process and displays the Applixware Main Menu.

See [CommonDlgClass Methods](#) for more information.

-> new_application@

Runs an application in a new task

Class CommonDlgClass set

Format this.new_application@(file[, arg1[, arg2[, arg3[, arg4[, arg5]]]])

Arguments file The Applixware Builder file, with full file path. If file is passed as an array, file[0] is used as the file name, and all other array items are taken as application arguments.

arg1 An optional application argument.

arg2 An optional application argument.

arg3 An optional application argument.
arg4 An optional application argument.
arg5 An optional application argument.

Description The set method `new_application@` runs an Applixware Builder application in a new task. Use the method to run multiple applications against the same axmain process. The data types of the optional arguments are defined by the target application.

See [CommonDlgClass Methods](#) for more information.

-> `object_server_close@`

Closes connection to object server

Class CommonDlgClass set

Format `this.object_server_close@(host)`

Arguments host The host machine.

Description The set method `object_server_close@` closes the connection to the object server. Call this method after removing objects with the [REMOTE OBJECT DESTROY@](#) macro.

See [CommonDlgClass Methods](#) for more information.

-> `object_server_open@`

Opens connection to object server

Class CommonDlgClass set

Format `this.object_server_open@(host)`

Arguments host The host machine.

Description The set method `object_server_open@` opens the connection to the object server. Call this method before creating objects with the [REMOTE OBJECT CREATE@](#) macro.

See [CommonDlgClass Methods](#) for more information.

-> **rt_disable_live_feed@**

Suspends interaction with the real-time server

Class CommonDlgClass set

Format this.rt_disable_live_feed@

Description The set method `rt_disable_live_feed@` suspends the current interaction with a real-time server. This method tells the gateway to stop sending information to the application invoking the method until it receives a [rt_enable_live_feed@](#) signal. The gateway continues to service other tasks.

Because the gateway is an independent process, it is possible to receive some data for up to a second or two after this macro executes

See [CommonDlgClass Methods](#) for more information.

-> **rt_display_status@**

Displays Real Time Status window

Class CommonDlgClass set

Format this.rt_display_status@

Description The set method `rt_display_status@` displays the Real Time Status window. Use the Real Time Status window provides information about the connection between the Real Time Engine and the data distribution system. Any information or error messages relating to the connection between the Real Time Engine and the data distribution system source display in the Real Time Status window.

See [CommonDlgClass Methods](#) for more information.

-> **rt_enable_live_feed@**

Restarts interaction with the real-time server

Class CommonDlgClass set

Format this.rt_enable_live_feed@

Description The set method `rt_enable_live_feed@` reenables a task's interaction with a real-time server. That is, the task will resume receiving data. Only the current task is affected; that is, if other tasks had disabled the feed of data, they will remain disabled.

See [CommonDlgClass Methods](#) for more information.

-> `rt_stop_restart_gateway_dlg@`

Displays the Stop or Restart Engine dialog box

Class CommonDlgClass set

Format `this.rt_stop_restart_gateway_dlg@`

Description The set method `rt_stop_restart_gateway_dlg@` displays the Stop or Restart Engine dialog box. Use the dialog box to close the connection between the application and the data distribution system without exiting the application or to re-establish connection to a data distribution system if that connection has been broken due to an error condition.

See [CommonDlgClass Methods](#) for more information.

-> `run_application_dlg@`

Displays Applixware Builder Run dialog box

Class CommonDlgClass set

Format `this.run_application_dlg@`

Description The set method `run_application_dlg@` displays the Applixware Builder Run dialog box. Use the dialog box to run multiple applications and macros.

See [CommonDlgClass Methods](#) for more information.

-> `run_macro@`

Runs a macro

Class CommonDlgClass set

Format `this.run_macro@(macro[, args])`

Arguments macro The macro name.
args An array of arguments for the macro.

Description The set method run_macro@ runs the passed macro. The macro must exist in your ELF search path.

See [CommonDlgClass Methods](#) for more information.

-> run_macro_dlg@

Displays Run Macro dialog box

Class CommonDlgClass set

Format this.run_macro_dlg@

Description The set method run_macro_dlg@ displays the Run Macro dialog box. Use the dialog box to prompt for a macro to run.

See [CommonDlgClass Methods](#) for more information.

-> send_mail_dlg@

Displays the Send Mail dialog box

Class CommonDlgClass set

Format this.send_mail_dlg@([format msg@ msg])

Arguments msg The message information, in msg@ format. The msg@ format is defined as follows:

format arrayof

msg_recips@ recips, Recipient information.

subject, The subject string.

body, An array of strings containing the message body.

afiles, An array of attached files.

attrs, The message attributes. The attributes are ORed together to set the attribute. The attributes are defined as:

ML#IS_URGENT Urgent

ML#IS_CERTIFIED Certified

ML#REPLY_RQST Reply requested
 ML#OBOX_COPY Copy to outbox
 reply_text, reply request text.
 The msg_recips@ format is defined as follows:
 recip_name, The recipient name
 recip_type The recipient types. The types are defined as:
 ML#TO_RECIP To recipient
 ML#CC_RECIP Carbon copy recipient
 ML#BC_RECIP Blind copy recipient

Description The set method send_mail_dlg@ displays the Send Mail dialog box. Use the dialog for a consistent mail interface. The message formats and variable definitions are defined in the builder_.am header file, located in your *install_dir/axdata/elf* directory, include this file when using the method.

See [CommonDlgClass Methods](#) for more information.

<- error_event

Called for system errors

Class CommonDlgClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [CommonDlgClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class CommonDlgClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [CommonDlgClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class CommonDlgClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [CommonDlgClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class CommonDlgClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [CommonDlgClass Methods](#) for more information.

<- invocation_mode@

Returns the invocation mode setting of a CORBA object

Class CorbaClass get

Format mode = this.invocation_mode@

Description The get method `invocation_mode@` returns the invocation mode setting of a CORBA object. The invocation mode can be one of the following values:

- Synchronous - If `invocation_mode@` = 1, the Builder application calls the method, then waits for the method call to be resolved. The application is blocked while the request is being processed. This is the default.
- One Way - If `invocation_mode@` = 2, the Builder application calls the method, then continues its own processing. This invocation method is used when the Application is not interested in the results of the method call.
- Deferred Synchronous - If `invocation_mode@` = 3, the Builder application calls the method, and receives a request ID. The application can then continue processing. At some point in the future, it calls the method `deferred_request_response@` to receive the results of the method call.

-> invocation_mode@

Sets the invocation mode setting of a CORBA object

Class CorbaClass Set

Format this.invocation_mode@(mode)

Arguments mode an integer. The mode should be either 1,2, or 3, depending on the invocation mode you are trying to establish:

Synchronous - If `invocation_mode@ = 1`, the Builder application calls the method, then waits for the method call to be resolved. The application is blocked while the request is being processed. This is the default.

One Way - If `invocation_mode@ = 2`, the Builder application calls the method, then continues its own processing. This invocation method is used when the Application is not interested in the results of the method call.

Deferred Synchronous - If `invocation_mode@ = 3`, the Builder application calls the method, and receives a request ID. The application can then continue processing. At some point in the future, you must call the method `this.deferred_request_response@` to receive the results of the method call.

Description Sets the `invocation_mode` of a CORBA object. If you set the invocation mode to Deferred Synchronous, you must call the method `this.deferred_request_response@` to complete the call.

`deferred_request_response@`

Returns the result of
a deferred method call

Class CorbaClass get

Format `this.deferred_request_response@(reqid)`

Arguments reqid This request id number is returned by a CorbaClass method when invocation mode = 3.

Description Returns the results of a Corba Class method called with `invocation_mode@ = 3` (deferred synchronous). The reqid must be the request id returned when the method was called. The `deferred_request_response@` method returns an array with the results of the method call.

<- cell_value_string@

Returns cross table value

Class CrossTableClass get

Format val = this.cell_value_string@

Description The get method cell_value_string@ returns the cross table value used to retrieve information. The value is a function, such as count(*), that is applied to the cross table column and row.

See [CrossTableClass Methods](#) for more information.

<- column_data_set_field_name@

Returns the data set column used for the cross table column

Class CrossTableClass get

Format field = this.column_data_set_field_name@

Description The get method column_data_set_field_name@ returns the data set column used for the cross table column values. The data set column is returned as a string. Use the method with the set method column_data_set_field_name@ to verify and change a cross table column.

See [CrossTableClass Methods](#) for more information.

<- create_totals_column@

Returns total column display status

Class CrossTableClass get

Format flag = this.create_totals_column@

Description The get method create_totals_column@ returns the total column display status in the cross table as a Boolean value. The method returns TRUE if the cross table contains a total column, otherwise it returns FALSE. Use the method with the set method create_totals_column@ to verify and change and the total column status.

See [CrossTableClass Methods](#) for more information.

<- create_totals_row@

Returns total row display status

Class CrossTableClass get

Format flag = this.create_totals_row@

Description The get method create_totals_row@ returns the total row displaystatus in the cross table as a Boolean value. The method returns TRUE if the cross table contains a total row, otherwise it returns FALSE. Use the method with the set method create_totals_row@ to verify and change the total row status.

See [CrossTableClass Methods](#) for more information.

<- is_column_name@

Returns validity of column name

Class CrossTableClass get

Format flag = this.is_column_name@

Description The get method is_column_name@ returns the validity of a column name as a Boolean value. The method returns TRUE if the column is a column in the data set, otherwise it returns FALSE. Use the method to verify a column name before using it in the cross table.

See [CrossTableClass Methods](#) for more information.

<- row_data_set_field_name@

Returns the data set column used for the cross table row

Class CrossTableClass get

Format field = this.row_data_set_field_name@

Description The get method `row_data_set_field_name@` returns the data set column used for the cross table row values. The data set column is returned as a string. Use the method with the set method `row_data_set_field_name@` to verify and change a cross table row.

See [CrossTableClass Methods](#) for more information.

-> `cell_value_string@`

Sets cross table value

Class CrossTableClass set

Format `this.cell_value_string@(value)`

Arguments value A string representing a function, such as `count(*)`.

Description The set method `cell_value_string@` sets the cross table value used to retrieve information. The value is a function, such as `count(*)`, that is applied to the cross table column and row. Use this method with the get method `cell_value_string@` to verify and change the cross table value.

See [CrossTableClass Methods](#) for more information.

-> `column_data_set_field_name@`

Sets the data set column used for the cross table column

Class CrossTableClass set

Format `this.column_data_set_field_name@(value)`

Arguments value A string representing the full data set column name.

Description The set method `column_data_set_field_name@` sets the data set column used for the cross table column. Use this method with the get method `column_data_set_field_name@` to verify and change the cross table column.

See [CrossTableClass Methods](#) for more information.

-> create_totals_column@

Sets the total column display status

Class CrossTableClass set

Format this.create_totals_column@(flag)

Arguments flag A Boolean value.

Description The set method create_totals_column@ sets the total column display status in the cross table. Pass the method TRUE add a total column to the cross table, otherwise pass the method FALSE to remove a total column from the cross table. Use this method with the get method create_totals_column@ to add or remove the cross table total column.

See [CrossTableClass Methods](#) for more information.

-> create_totals_row@

Sets the total row display status

Class CrossTableClass set

Format this.create_totals_row@(flag)

Arguments flag A Boolean value.

Description The set method create_totals_row@ sets the total row display status in the cross table. Pass the method TRUE add a total row to the cross table, otherwise pass the method FALSE to remove a total row from the cross table. Use this method with the get method create_totals_row@ to add or remove the cross table total row.

See [CrossTableClass Methods](#) for more information.

-> row_data_set_field_name@

Sets the data set column used for the cross table row

Class CrossTableClass set

Format this.row_data_set_field_name@(value)

Arguments value A string representing the full data set column name.

Description The set method `row_data_set_field_name@` sets the data set column used for the cross table row. Use this method with the get method `row_data_set_field_name@` to verify and change the cross table row.

See [CrossTableClass Methods](#) for more information.

-> selection_changed_event

Called when cross table selection changes

Class CrossTableClass event

Format `this.selection_changed_event(rowArray)`

Arguments rowArray An array of row numbers. Rows are zero based.

Description The `selection_changed_event` is called by Applixware Builder when the selections in the cross table change when the cross table is not editable. This is a user-defined event, place all actions you want performed in the event definition.

See [CrossTableClass Methods](#) for more information.

<- data_set@

Returns data set used by object

Class ControlClass get

Format `object data = this.data_set@`

Description The get method `data_set@` returns the data set used by the object.

See [ControlClass Methods](#) for more information.

<- data_set_field_name@

Returns data set field used by object

Class ControlClass get

Format field = this.data_set_field_name@

Description The get method data_set_field_name@ returns the data set field used by the object. See [ControlClass Methods](#) for more information.

<- data_set_field_name_list@

Returns data set fields used by object

Class ControlClass get

Format fieldArray = this.data_set_field_name_list@

Description The get method data_set_field_name_list@ returns the data set fields used by the object as an array.

See [ControlClass Methods](#) for more information.

<- data_set_field_value@

Returns the current data set field value of the object

Class ControlClass get

Format value = this.data_set_field_value@

Description The get method data_set_field_value@ returns the current data set field value of the object.

See [ControlClass Methods](#) for more information.

<- data_source@

Returns the data source object

Class ControlClass get

Format object source = this.data_source@

Description The get method data_source@ returns the data source object associated with the dialog box control. The method returns NULL if no data source is associated with the dialog box control.

See [ControlClass Methods](#) for more information.

<- data_source_is_connected@

Indicates if data source is available

Class ControlClass get

Format flag = this.data_source_is_connected@

Description The get method data_source_is_connected@ indicates if a data source is available. The method returns TRUE if a data set, Real Time, or macro data source is connected to the object. The method returns FALSE if no data source is connected to the object.

See [ControlClass Methods](#) for more information.

<- data_source_macro@

Returns the data source macro for the object

Class ControlClass get

Format name = this.data_source_macro@

Description The get method data_source_macro@ returns the name of the data source macro for the object.

See [ControlClass Methods](#) for more information.

<- data_source_type@

Returns the data source type associated with the object

Class ControlClass get

Format name = this.data_source_type@

Description The get method data_source_type@ returns the data source type associated with the object. The valid data source types are:

- 0 None
- 1 Macro
- 2 Data set
- 3 Real Time

See [ControlClass Methods](#) for more information.

<- data_source_value@

Returns the current data source value of the object

Class ControlClass get

Format value = this.data_source_value@

Description The get method data_source_value@ returns the data source value of the object.

See [ControlClass Methods](#) for more information.

<- display_map_macro@

Returns the display map macro for the object

Class ControlClass get

Format name = this.display_map_macro@

Description The get method display_map_macro@ returns the display map macro for the object. A display map is a macro or method to manipulate information retrieved from a data source before it is displayed in the object.

See [ControlClass Methods](#) for more information.

<- display_suspend@

Returns the suspended state of the object

Class ControlClass get

Format flag = this.display_suspend@

Description The get method display_suspend@ returns the suspended state of the object. If the method returns TRUE the display of object information is suspended. If the method returns FALSE updated information is displayed in the object.

See [ControlClass Methods](#) for more information.

<- is_grayed@

Indicates enable state of object

Class ControlClass get

Format flag = this.is_grayed@

Description The get method is_grayed returns a Boolean value indicating the enable state of the object. The method returns TRUE if the object is grayed and unavailable, otherwise it returns FALSE, indicating the object is ungrayed and available. You can use this method with the set method is_grayed to verify and change the object's grayed state.

For example, to make the object ungrayed you would use the method as follows:

```
var flag
flag = this.is_grayed           'get method
IF flag
    this.is_grayed = FALSE     'set method
```

See [ControlClass Methods](#) for more information.

<- is_hidden@

Indicates hidden state of object

Class ControlClass get

Format flag = this.is_hidden@

Description The get method is_hidden@ returns a Boolean value indicating the hidden state of the object. The method returns TRUE if the object is hidden and not visible in the dialog box, otherwise it returns FALSE, indicating the object is not hidden and is visible in the dialog box. You can use this method with the set method is_hidden@ to verify and change the object's hidden state.

For example, to make the object hidden you would use the method as follows:

```
var flag
flag = this.is_hidden@           'get method
IF NOT flag
    this.is_hidden@ = TRUE     'set method
```

See [ControlClass Methods](#) for more information.

<- rt_field_name@

Returns the Real Time field associated with the object

Class ControlClass get

Format name = this.rt_field_name@

Description The get method rt_field_name@ returns the Real Time record field associated with the object.

See [ControlClass Methods](#) for more information.

<- rt_field_value@

Returns the current Real Time field value for the object

Class ControlClass get

Format value = this.rt_field_value@

Description The get method rt_field_value@ returns the current Real Time record field value for the object.

See [ControlClass Methods](#) for more information.

<- rt_gateway@

Returns the Real Time gateway associated with the object

Class ControlClass get

Format object gate = this.rt_gateway@

Description The get method rt_gateway@ returns the Real Time gateway object associated with the object.

See [ControlClass Methods](#) for more information.

<- rt_record_name@

Returns the Real Time record associated with the object

Class ControlClass get

Format name = this.rt_record_name@

Description The get method rt_record_name@ returns the Real Time record associated with the object.

See [ControlClass Methods](#) for more information.

<- title@

Returns the object title

Class ControlClass get

Format name = this.title@

Description The get method `title@` returns the object title as a string. You can use this method with the set method `title@` to verify and change the object's title. The object's title is the name used to reference the object in the application.

See [ControlClass Methods](#) for more information.

`<- validator_macro@`

Returns the validator macro for the object

Class ControlClass get

Format `name = this.validator_macro@`

Description The get method `display_map_macro@` returns the validator macro for the object. A validator is a macro or method to verify the information in the object before it is written to a database.

See [ControlClass Methods](#) for more information.

`<- x_pos@`

Returns object x-axis position in dialog box

Class ControlClass get

Format `position = this.x_pos@`

Description The get method `x_pos@` returns the x-axis position, in pixels, of the object's top left corner. The position is relative to the top left corner of the dialog box, position 0. You can use this method with the get method `y_pos@` to get the object's location in the dialog box.

For example, to set the object text size to use the following:

```
var x, y
x = this.x_pos@
y = this.y_pos@
```

See [ControlClass Methods](#) for more information.

<- y_pos@

Returns object y-axis position in dialog box

Class ControlClass get

Format position = this.y_pos@

Description The get method y_pos@ returns the y-axis position, in pixels, of the object's top left corner. The position is relative to the top left corner of the dialog box, position 0. You can use this method with the get method x_pos@ to get the object's location in the dialog box.

For example, to set the object text size to you would use the method as follows:

```
var x, y
x = this.x_pos@
y = this.y_pos@
```

See [ControlClass Methods](#) for more information.

-> data_set@

Sets data set used by object

Class ControlClass set

Format this.data_set@(object data)

Arguments data A data set data object.

Description The set method data_set@ sets the data set used by the object.

See [ControlClass Methods](#) for more information.

-> data_set_field_name@

Sets data set field used by object

Class ControlClass set

Format this.data_set_field_name@(field)

Arguments field A field name in the data set object.

Description The set method `data_set_field_name@` sets the data set field used by the object.
See [ControlClass Methods](#) for more information.

-> `data_set_field_name_list@`

Sets data set fields used by object

Class ControlClass set

Format `this.data_set_field_name_list@(fieldArray)`

Arguments fieldArray An array of field names in the data set object.

Description The set method `data_set_field_name_list@` sets the data set fields used by the object as an array.

See [ControlClass Methods](#) for more information.

-> `data_set_field_value@`

Sets the current data set field value of the object

Class ControlClass set

Format `this.data_set_field_value@(value)`

Arguments value A valid field value.

Description The set method `data_set_field_value@` sets the current data set field value of the object.

See [ControlClass Methods](#) for more information.

-> `data_source_macro@`

Sets the data source macro for the object

Class ControlClass set

Format this.data_source_macro@(name)

Arguments name A macro name.

Description The set method data_source_macro@ sets the name of the data source macro for the object.

See [ControlClass Methods](#) for more information.

-> data_source_type@

Sets the data source type associated with the object

Class ControlClass set

Format this.data_source_type@(type)

Arguments type One of the valid source types, as defined in the header file builder_.am, located in the *install_dir/axdata/elf* directory. The valid source types are:

DATA_SOURCE#NONE	None
DATA_SOURCE#MACRO	Macro
DATA_SOURCE#DATA_SET	Data set
DATA_SOURCE#REALTIME	Real Time

Description The set method data_source_type@ sets the data source type associated with the object.

See [ControlClass Methods](#) for more information.

-> data_source_value@

Sets the current data source value of the object

Class ControlClass set

Format this.data_source_value@

Arguments value A valid data source value.

Description The set method data_source_value@ sets the data source value of the object.

See [ControlClass Methods](#) for more information.

-> display@

Displays the object

Class ControlClass set

Format this.display@

Description The set method display@ displays the object in the dialog box. Use the method to update the display of an object after suspending the display with the set method display_suspend@. Use the methods to update the dialog box widgets at one time, instead of individually.

See [ControlClass Methods](#) for more information.

-> display_map_macro@

Sets the display map macro for the object

Class ControlClass set

Format this.display_map_macro@(name)

Arguments name A macro or method name.

Description The set method display_map_macro@ sets the display map macro for the object. A display map is a macro or method to manipulate information retrieved from a data source before it is displayed in the object. Macro names must be preceded by the @ character, otherwise the display map is considered a get method.

See [ControlClass Methods](#) for more information.

-> display_suspend@

Sets the suspended state of the object

Class ControlClass set

Format this.display_suspend@(flag)

Arguments flag A Boolean value. TRUE suspends the display of object information. FALSE displays updated information in the object.

Description The set method `display_suspend@` sets the suspended state of the object.

See [ControlClass Methods](#) for more information.

-> `is_grayed@`

Sets enable state of object

Class ControlClass set

Format `this.is_grayed@(flag)`

Arguments flag Indicates the enable state of the object. TRUE grays the object, making it unavailable, FALSE ungrays the object, making it available.

Description The set method `is_grayed@` sets the enable state of the object. You can use this method with the get method `is_grayed@` to verify and change the object's grayed state.

For example, to make the object ungrayed you would use the method as follows:

```
var flag
flag = this.is_grayed@           'get method
IF flag
    this.is_grayed@ = FALSE     'set method
```

See [ControlClass Methods](#) for more information.

-> `is_hidden@`

Sets hidden state of object

Class ControlClass set

Format `this.is_hidden@(flag)`

Arguments flag Indicates the hidden state of the object. TRUE hides the object, FALSE unhides the object, making it visible in the dialog box.

Description The set method `is_hidden@` sets the hidden state of the object. You can use this method with the get method `is_hidden@` to verify and change the object's hidden state.

For example, to make the object hidden you would use the method as follows:

```
var flag
```

flag = this.is_hidden@ 'get method

IF NOT flag

this.is_hidden@ = TRUE 'set method

See [ControlClass Methods](#) for more information.

-> rt_field_name@

Sets the Real Time field associated with the object

Class ControlClass set

Format this.rt_field_name@(name)

Arguments name A record field name.

Description The set method rt_field_name@ sets the Real Time record field associated with the object.

See [ControlClass Methods](#) for more information.

-> rt_field_value@

Sets the current Real Time field value for the object

Class ControlClass set

Format this.rt_field_value@(value)

Arguments value A valid record field value.

Description The set method rt_field_value@ sets the current Real Time record field value for the object.

See [ControlClass Methods](#) for more information.

-> rt_gateway@

Sets the Real Time gateway associated with the object

Class ControlClass set

Format this.rt_gateway@(object gate)

Arguments gate A Real Time gateway object.

Description The set method rt_gateway@ sets the Real Time gateway object associated with the object.

See [ControlClass Methods](#) for more information.

-> rt_record_name@

Sets the Real Time record associated with the object

Class ControlClass set

Format this.rt_record_name@(name)

Arguments name A valid record name.

Description The set method rt_record_name@ sets the Real Time record associated with the object.

See [ControlClass Methods](#) for more information.

-> title@

Sets the object title

Class ControlClass set

Format this.title@(name)

Arguments name A title string.

Description The set method title@ sets the object title to the passed string. You can use this method with the get method title@ to verify and change the object's title. The object's title is the name used to reference the object in the application.

See [ControlClass Methods](#) for more information.

-> validator_macro@

Sets the validator macro for the object

Class ControlClass set

Format this.validator_macro@(name)

Arguments name A macro or method name.

Description The set method validator_macro@ sets the validator macro for the object. A validator is a macro or method to verify the information in the object before it is written to a data-base. Macro names must be preceded by the @ character, otherwise the validator is considered a get method.

See [ControlClass Methods](#) for more information.

-> x_pos@

Sets object x-axis position in dialog box

Class ControlClass set

Format this.x_pos@(position)

Arguments position The x-axis position, in pixels, of the top left corner of the object dialog box. The position is relative to the top left corner of the dialog box, position 0.

Description The set method x_pos@ sets the object's x-axis position in the dialog box. You can use this method with the set method y_pos@ to set an object position.

For example, to set an object to display at position (10,50) you would call the method as:

```
    this.x_pos@(10)
```

```
    this.y_pos@(50)
```

See [ControlClass Methods](#) for more information.

-> y_pos@

Sets object y-axis position in dialog box

Class ControlClass set

Format this.y_pos@(position)

Arguments position The y-axis position, in pixels, of the top left corner of the object dialog box. The position is relative to the top left corner of the dialog box, position 0.

Description The set method y_pos@ sets the object's y-axis position in the dialog box. You can use this method with the set method x_pos@ to set an object position.

For example, to set a dialog box to display at position (10,50) you would call the method as:

```
    this.x_pos@(10)
```

```
    this.y_pos@(50)
```

See [ControlClass Methods](#) for more information.

<- error_event

Called for system errors

Class ControlClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ControlClass Methods](#) for more information.

-> **resize_event**

Called when a dialog box is resized

Class ControlClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [ControlClass Methods](#) for more information.

-> **initialize_event**

Called before displaying a control

Class ControlClass event

Format this.initialize_event

Description The `initialize_event` is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ControlClass Methods](#) for more information.

-> **terminate_event**

Called before closing or destroying dialog box

Class ControlClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ControlClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class ControlClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [ControlClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class ControlClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [ControlClass Methods](#) for more information.

<- all_values@

Returns data set information

Class DatasetClass get

Format infoArray = this.all_values@([colNames])

Arguments colNames An array of column names.

Description The get method all_values@ returns the data set information as a two-dimensional array. Use this method to retrieve data set information in a format ready to place in a dialog box table. The first dimension defines the row, the second dimension defines each column in the row.

If an array of column names is passed as an argument, then the information for the columns is returned in each row. If no argument is passed, then the information for displayed columns is returned.

See [DatasetClass Methods](#) for more information.

<- auto_query@

Returns auto-query status

Class DatasetClass get

Format flag = this.auto_query@

Description The get method auto_query@ returns the auto-query status as a Boolean value. The method returns TRUE if auto-query is enabled, otherwise it returns FALSE. If auto-query is enabled, the database is re-queried on any change in the query information. If auto-query is not enabled, the database is only re-queried when the set method requery_database@ is used.

See [DatasetClass Methods](#) for more information.

<- binary_size_limit@

Returns maximum data size

Class DatasetClass get

Format size = this.binary_size_limit@

Description The get method binary_size_limit@ returns the maximum size of binary data, in bytes, allowed in the database. This value is database-dependent.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- columns_in_table@

Returns table column information

Class DatasetClass get

Format format col_info@ columns = this.columns_in_table@(table)

Arguments table A table in the current database.

Description The get method columns_in_table@ returns database table column information as an array. The array contains the following information:

columns.names[i]	The actual table column name in the DBMS.
columns.types[i]	The data type. The valid data types are:
1	Character
2	Number
3	Binary
4	Date/time
5	Currency
6	Date

The col_info@ format and data types are defined in the dbase_.am file located in the install_dir/axdata/elf directory.

See [DatasetClass Methods](#) for more information.

<- command_object@

Returns SQLCommandClass object

Class DatasetClass get

Format obj = this.command_object@

Description The get method `command_object@` returns the `SQLCommandClass` object associated with this dataset. Use this method to access methods that give you additional control over the data you are manipulating.

See [DatasetClass Methods](#) for more information.

<- conditions@

Returns current query condition

Class DatasetClass get

Format format arrayof [sql_condition_info@](#) conditions = this.conditions@

Description The get method `conditions@` returns the query condition that is part of the current Data query. A Data query is made up of one or more conditions. The structure `sql_condition_info@`, defined in the header file `dbase_.am` located in the `/install_dir/axdata/elf` directory, contains the following information:

format `sql_condition_info@`

<code>table1,</code>	The table in the database containing column1
<code>column1,</code>	The comparison column for the condition
<code>compare,</code>	(Not Used)
<code>operate,</code>	The comparison operator. The valid comparison operators are:
	< Less than
	<= Less than or equal
	= Equal (basic operator)
	>= Greater than or equal
	> Greater than
	In In (basic operator)
	Between Between (basic operator)
	Like Like (basic operator)
	Is Null Is Null (basic operator)
<code>not_it,</code>	Boolean value, if TRUE means use inverse of operator
<code>value,</code>	The constant value for a Value query
<code>valueList,</code>	A list of values for "In" or "Between" query condition
<code>table2,</code>	The table in the database containing column2
<code>column2,</code>	The column used for a Column query condition
<code>or_mode,</code>	Boolean value, where TRUE means OR, FALSE means AND. For use with multiple conditions, defines relation of this condition to previous condition
<code>unused2,</code>	(Not Used)

sqlstr,	A subselect string for a Subselect query
type,	Type of query: "Value", "Column", or "Subselect"
front_parens,	The number of left parentheses before the condition, for multiple conditions
rear_parens,	The number of right parentheses after the condition, for multiple conditions

See [DatasetClass Methods](#) for more information.

<- connection@

Returns database connection info

Class DatasetClass get

Format format sql_dbase_entry@ connect = this.connection@

Description The get method connection@ returns the database connection information for the query. The structure sql_dbase_entry@, defined in the header file dbase_.am located in the *install_dir*/axdata/elf directory, contains the following information:

format sql_dbase_entry@	
vendor,	The database vendor name
database,	The database name
host,	The host machine for the database
server,	The database SQL server name
routing,	The vendor routing name.
port,	NULL
serviceName	The axnet service name to ELF/SQL server

See [DatasetClass Methods](#) for more information.

<- connection_info@

Returns database connection info

Class DatasetClass get

Format format sql_dbase_entry@ connect = this.connection_info@

Description The get method connection_info@ returns the database connection information for the query. The structure sql_dbase_entry@, defined in the header file dbase_.am located in the *install_dir*/axdata/elf directory, contains the following information:

format sql_dbase_entry@	
vendor,	The database vendor name
database,	The database name
host,	The host machine for the database
server,	The database SQL server name
routing,	The vendor routing name.
port,	NULL
serviceName	The axnet service name to ELF/SQL server

See [DatasetClass Methods](#) for more information.

<- connection_object@

Returns SQLConnectClass object

Class DatasetClass get

Format obj = this.connection_object@

Description The get method connection_object@ returns the SQLConnectClass object associated with this dataset. Use this method to access methods that give you additional control over the data you are manipulating.

See [DatasetClass Methods](#) for more information.

<- cursor_props@

Returns data set cursor properties

Class DatasetClass get

Format format sql_cursor_properties@ props = this.cursor_props@

Description The get method cursor_props@ returns the data set cursor properties. The structure sql_cursor_properties@, defined in the header file dbase_.am located in the */install_dir/axdata/elf* directory, contains the following information:

format sql_cursor_properties@	
fetch_size,	The quantity of records (rows) to retrieve in a query of an Oracle® database server. This is an internal preference to potentially increase query speed, and does not have a visible effect on the number of records returned by the query. The default value is 50.

trim_strings,	A value, as defined in the header file <code>dbase_.am</code> , located in the <code>install_dir/axdata/elf</code> directory. The defined values are: SQL_TRIM_BOTH_ Remove blank space from beginning and end of string. SQL_TRIM_BEGIN_ Remove blank space from beginning of string. SQL_TRIM_END_ Remove blank space from end of string. SQL_TRIM_NONE_ Leave string as is. The default value is <code>SQL_TRIM_BOTH_</code> .
transpose_data,	A Boolean value. TRUE transposes the columns and rows of the returned information. FALSE does not change the returned data. The default value is FALSE.
nums_as_nums,	A Boolean value. TRUE signals Applixware Builder to handle numeric information in number format. FALSE signals Applixware Builder to handle numeric information as strings. The default value is FALSE.
binary_size_limit,	The maximum size, in bytes, of binary data. This value is database-dependent.
timeout,	The query timeout, in seconds. Use 0 for infinite limit, use NULL for default limit.
max_rows,	The maximum quantity of records to retrieve. Pass NULL to get the default DBMS quantity.
dates_as_dates	A Boolean value. TRUE signals Applixware Builder to handle date information in date format. FALSE signals Applixware Builder to handle date information as strings. The default value is FALSE.

See [DatasetClass Methods](#) for more information.

<- current_record_value@

Returns record value for column

Class DatasetClass get

Format value = this.current_record_value@(name)

Arguments name The column name in the current table.

Description The get method `current_record_value@` returns a column record value for the currently censored record.

For example, to get a column record value for the column Name use the following:

var value

value = this.current_record_value@("Name")

See [DatasetClass Methods](#) for more information.

<- datatype@

Returns column data type

Class DatasetClass get

Format values = this.datatype@(colNum)

Arguments colNum The column number. Columns are 0-based.

Description The get method `datatype@` returns the column data type. The valid data types are:

- 1 Character
- 2 Number
- 3 Binary
- 4 Date/time
- 5 Currency
- 6 Date

See [DatasetClass Methods](#) for more information.

<- dates_as_dates@

Returns date handling

Class DatasetClass get

Format format = this.dates_as_dates@

Description The get method `dates_as_dates@` returns the date handling procedure for date information. TRUE means Applixware Builder handles date information in date format. FALSE means Applixware Builder handles date information as strings.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- date_format@

Returns date/time format

Class DatasetClass get

Format format = this.date_format@

Description The get method `date_format@` returns the date/time format of dates and time values in returned by the query. The date/time formats are defined in the `datetim_.am` header file located in the `/install_dir/axdata/elf` directory.

See [DatasetClass Methods](#) for more information.

<- displayed_columns@

Returns query column names

Class DatasetClass get

Format columns = this.displayed_columns@

Description The get method `displayed_columns@` returns an array of columns in the data set query. For example, to get the columns in the current data set query use the following:

```
var columns
columns = this.displayed_columns@
```

See [DatasetClass Methods](#) for more information.

<- display_binary_data@

NA

This method does not exist, use the set method [display_binary_data@](#) to display binary data.

See [DatasetClass Methods](#) for more information.

<- exec_direct@

Executes an SQL statement

Class DatasetClass get

Format info = this.exec_direct@(sqlStmt)

Arguments sqlStmt The command being executed; sql_stmt can be a string or an array of strings.

Description Executes the passed statement directly and returns the results. No explicit *prepare* statements or cursor calls need be used.

If sqlStmt is not a query statement, ELF SQL performs the following steps:

1. Prepares the statement.
2. Executes the statement.
3. Unprepares the statement.

Any data that would be returned by sqlStmt is returned.

If the statement is a query statement, the following sequence of operations occurs:

1. Prepares the statement.
2. Declares the cursor.
3. Opens the cursor.
4. Fetches all of the data.
5. Closes the cursor.
6. Unprepares the statement.

While it is very easy to invoke SQL statements using the `exec_direct@` method, it can greatly increase your overhead because of the added expense of repeatedly preparing the statement.

The information that is returned by this method varies and is dependent upon the statement that is executed.

See [DatasetClass Methods](#) for more information.

<- expressions@

Returns query expressions

Class DatasetClass get

Format columns = this.expressions@

Description The get method `expressions@` returns expressions in the database query as an array of strings. An expression is a query column created by using SQL functions with the columns from your database query. The functions include aggregate functions, such as sum (`sum`), average (`avg`), maximum (`max`), and minimum (`min`).

See [DatasetClass Methods](#) for more information.

<- fetch_size@

Returns number of records to retrieve

Class DatasetClass get

Format value = this.fetch_size@

Description The get method `fetch_size@` returns the number of records (rows) to retrieve in a query of an Oracle® database server. This is an internal preference to potentially increase query speed, and does not have a visible effect on the number of records returned by the query.

See [DatasetClass Methods](#) for more information.

<- get_cursor@

NOTE: This is an obsolete method, use [position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- get_position@

Returns current record number

Class DatasetClass get

Format value = this.get_position@

Description The get method `get_position@` returns a numeric value for the current record number. The record numbers are 0-based.

For example, to get the record number of the current record use the following:

```
var value
value = this.get_position@ 'Get current record number
value = value +1    'Set to next record
this.set_position@(value)
```

NOTE: This is an obsolete method, use [position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- group_by@

Returns "group by" columns

Class DatasetClass get

Format groupByColumnsArray = this.group_by@

Description The get method `group_by@` returns the "group by" columns in the current database query as an array of strings.

See [DatasetClass Methods](#) for more information.

<- has_more_records@

Indicates if additional records meet query conditions

Class DatasetClass get

Format flag = this.has_more_records@

Description The get method has_more_records@ indicates if additional records that meet query conditions haven't been fetched. The method returns a Boolean value. TRUE means additional records can be fetched, FALSE means all records have been fetched.

See [DatasetClass Methods](#) for more information.

<- having@

Returns "having" conditions

Class DatasetClass get

Format format arrayof [sql_condition_info@](#) conditions = this.having@

Description The get method having@ returns a structure containing the query "having" conditions for the current database query. A "having" database query is made up of one or more "having" conditions. A "having" database query is made against the set of "group by" columns. The structure sql_condition_info@, defined in the header file dbase_.am located in the *install_dir*/axdata/elf directory, contains the following information:

format sql_condition_info@

table1,	The table in the database containing column1
column1,	The comparison column for the condition
compare,	(Not Used)
operate,	The comparison operator. The valid comparison operators are:
	< Less than
	<= Less than or equal
	= Equal (basic operator)
	>= Greater than or equal
	> Greater than
	In In (basic operator)
	Between Between (basic operator)
	Like Like (basic operator)
	Is Null Is Null (basic operator)
not_it,	Boolean value, if TRUE means use inverse of operator
value,	The constant value for a Value query
valueList,	A list of values for "In" or "Between" query condition
table2,	The table in the database containing column2
column2,	The column used for a Column query condition
or_mode,	Boolean value, where TRUE means OR, FALSE means AND. For use with multiple conditions, defines relation of this condition to previous condition

unused2,	(Not Used)
sqlstr,	A subselect string for a Subselect query
type,	Type of query: "Value", "Column", or "Subselect"
front_parens,	The number of left parentheses before the condition, for multiple conditions
rear_parens,	The number of right parentheses after the condition, for multiple conditions

See [DatasetClass Methods](#) for more information.

<- is_editable_by_default@

Returns initial editing status

Class DatasetClass get

Format flag = this.is_editable_by_default@

Description The get method `is_editable_by_default@` returns the initial editing status of a data set. The method returns TRUE if the database editing is set by default. The method returns FALSE if the database editing is not on by default.

See [DatasetClass Methods](#) for more information.

<- is_edited@

Returns query edit status

Class DatasetClass get

Format flag = this.is_edited@

Description The get method `is_edited@` returns the query edit status as a Boolean value. The method returns TRUE if the current query is edited, otherwise it returns FALSE.

See [DatasetClass Methods](#) for more information.

<- is_editing_enabled@

Returns editing enabled status

Class DatasetClass get

Format flag = this.is_editing_enabled@

Description The get method is_editing_enabled@ returns the editing enabled status as a Boolean. The method returns TRUE if query editing is enabled, otherwise it returns FALSE.

See [DatasetClass Methods](#) for more information.

<- is_transactionless@

Returns database transaction status

Class DatasetClass get

Format flag = this.is_transactionless@

Description The get method is_transactionless@ returns the database transaction status. The method returns TRUE if the database does not have transactions. The method returns FALSE if the database has transactions.

See [DatasetClass Methods](#) for more information.

<- join@

Returns join information

Class DatasetClass get

Format format arrayof sql_join_info@ join = this.join@

Description The get method join@ returns an array of structures containing join information for the current database query. The structure sql_join_info@, defined in the header file dbase_.am located in the */install_dir /axdata/elf* directory, contains the following attributes:

format sql_join_info@

table1,	Name of the first table
table2,	Name of the second table
column1,	Name of the join column in table1
column2,	Name of the join column in table2
outer1,	Boolean value where TRUE means a right outer join
outer2,	Boolean value where TRUE means a left outer join
operate,	One of the following comparison operators:
=	equal

!= not equal
< less than
<= less than or equal
> greater than
>= greater than or equal

See [DatasetClass Methods](#) for more information.

<- join_info@

Returns join information

This method is obsolete, use the [join@](#) method to get table join information.

<- lock_on_edit@

Returns row locking operation

Class DatasetClass get

Format flag = this.lock_on_edit@

Description The get method lock_on_edit@ returns row locking operation as a Boolean value. FALSE does not lock on edit, TRUE locks on edit. The default is TRUE.

See [DatasetClass Methods](#) for more information.

<- login@

Returns database login info

Class DatasetClass get

Format format login_info_format@ logArray = this.login@(flag)

Arguments flag A Boolean value. If flag is set to TRUE, the method returns the login information as is. If flag is set to FALSE, the method displays the Login Identification dialog box if the user name and password settings are not initialized.

Description The get method `login@` returns database login information for the current data set in `login_info_format@`. The format is defined in the header file `sqlc_.am`, located in the `install_dir/axdata/elf` directory. The format is defined as follows:

`format login_info_format@`

<code>username,</code>	<code>user name</code>
<code>passwd,</code>	<code>user password</code>
<code>is_ready</code>	Boolean value, TRUE means user name and passwordd initialized

`username` is the database user name, `passwd` is the user password, and `is_ready` is the initialization state of the user name and password. A TRUE value for `is_ready` means the user name and password are initialized. The initialization state helps you determine if a user name or password were deliberately set to NULL, or if they are not set.

See [DatasetClass Methods](#) for more information.

`<- login_dlg@`

Displays login dialog box

Class `DatasetClass` get

Format `logArray = this.login_dlg@`

Description The get method `login_dlg@` displays the Login Identification dialog box. Use the dialog box as a consistent user interface for setting the user name and password for database login.

If new login information is entered in the dialog box and the OK button is pressed, the method returns the new database login information as an array. `logArray[0]` is the database user name, `logArray[1]` is the user password, and `logArray[2]` is the initialization state of the user name and password. A TRUE value for `logArray[2]` means the user name and password are initialized. The initialization state helps you determine if a user name or password were deliberately set to NULL, or if they are not set. If the dialog box is canceled the previous login settings are returned.

See [DatasetClass Methods](#) for more information.

`<- login_global@`

Returns database login info

Class `DatasetClass` get

Format logArray = this.login_global@

Description The set method login_global@ returns the user name and password for all data sets for database login. logArray[0] is the database user name, logArray[1] is the user password. All data sets use this information by default, unless login information is set for the specific data set with the [login@](#) method

See [DatasetClass Methods](#) for more information.

<- max_records@

Returns maximum quantity of records

Class DatasetClass get

Format num = this.max_records@

Description The get method max_records@ returns the maximum quantity of records retrieved by a query. Limiting the number of records you retrieve may reduce the time spent executing a query.

See [DatasetClass Methods](#) for more information.

<- no_duplicates@

Returns "allow duplicates" status

Class DatasetClass get

Format flag = this.no_duplicates@

Description The get method no_duplicates@ returns the "allow duplicates" status of the query as a Boolean value. FALSE allows duplicate rows in the query, TRUE does not allow duplicate rows in the query.

See [DatasetClass Methods](#) for more information.

<- numbers_as_numbers@

Returns number handling

Class DatasetClass get

Format flag = this.numbers_as_numbers@

Description The get method numbers_as_numbers@ returns the default handling procedure for numeric information as a Boolean value. TRUE signals Applixware Builder to handle numeric information in number format. FALSE signals Applixware Builder to handle numeric information as strings. Numbers are converted to strings by default.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- order_by@

Returns sort order

Class DatasetClass get

Format format arrayof sql_sort_info@ order = this.order_by@

Description The get method order_by@ returns the sort order information in the data set query. The structure sql_sort_info@, defined in the header file dbase_.am located in the *install_dir /axdata/elf* directory, contains the following information:

format sql_sort_info@

column_name, The column name as a string.

descending A Boolean value, where TRUE means descending sort, and FALSE means ascending sort.

See [DatasetClass Methods](#) for more information.

<- passwd@

Returns database password

Class DatasetClass get

Format password = this.passwd@

Description The get method passwd@ returns the password used to connect to the database.

See [DatasetClass Methods](#) for more information.

<- position@

Returns current position

Class DatasetClass set

Format val = this.position@

Description The get method position@ returns the database position as a numeric value. Records are 0-based.

For example, to get the current position and set the position to the next record you would call the method as:

```
var value
value = this.position@    'Get current record number
value = value +1        'Set to next record
this.position@(value)
```

See [DatasetClass Methods](#) for more information.

<- record_count@

Returns quantity of records retrieved

Class DatasetClass get

Format value = this.record_count@

Description The get method record_count@ returns the quantity of records currently retrieved by the data set query.

For example, to get the quantity of records currently retrieved by the data set query use the following:

```
var value
value = this.record_count@
```

See [DatasetClass Methods](#) for more information.

<- sort_info@

Returns sort information

Class DatasetClass get

Format format arrayof sql_sort_info@ sort = this.sort_info@

Description The get method `sort_info@` returns an array of structures containing sort information for the current database query. The structure `sql_sort_info@`, defined in the header file `dbase_.am` located in the `/install_dir /axdata/elf` directory, contains the following information:

format `sql_sort_info@`

column_name,	The name of the column determining the sort
descending	Boolean value where TRUE means descending sort, and FALSE means ascending sort

NOTE: This is an obsolete method, use [order_by@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- source@

Returns SQL source code

Class DatasetClass get

Format value = this.source@

Description The get method `source@` returns an array of strings containing the SQL source code generated for the current database query.

See [DatasetClass Methods](#) for more information.

<- sql_heading@

Returns dataset heading information

Class DatasetClass get

Format format arrayof sql_heading@ headers = this.sql_heading@

Description The get method sql_heading@ returns an array of table heading information for the dataset in sql_heading@ format. The sql_heading@ format is defined in the dbase_.am file located in the install_dir/axdata/elf directory..

See [DatasetClass Methods](#) for more information.

<- tables@

Returns table information

Class DatasetClass get

Format format arrayof sql_table_info@ sort = this.tables@

Description The get method tables@ returns a structure containing information about tables used in the current database query. The structure sql_table_info@, defined in the header file dbase_.am located in the *install_dir/axdata/elf* directory, contains the following information:

format sql_table_info@
name, The informal table name
uid, The fully qualified, formal table name
colnames The array of column names

See [DatasetClass Methods](#) for more information.

<- tables_used@

This method is obsolete, use the get method [tables@](#) to get table information used in the query.

<- timeout@

Returns query timeout limit

Class DatasetClass get

Format time = this.timeout@

Description The get method timeout@ returns the query timeout limit, in seconds. The method limits the duration of query. Use the timeout to avoid time consuming queries, or to cancel queries that would continue indefinitely.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- transpose_data@

Returns data transpose status

Class DatasetClass get

Format flag = this.transpose_data@

Description The get method transpose_data@ returns the data transpose status as a Boolean value. TRUE transposes the columns and rows of the returned information. FALSE does not change the returned data. For example, if the method is set to TRUE the information originally retrieved as data[2,1] becomes data[1,2].

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- trim_strings@

Returns blank space handling in strings

Class DatasetClass get

Format value = this.trim_strings@

Description The get method trim_strings@ returns the blank space handling of returned strings. The values are defined in the header file dbase_.am, located in the *install_dir/axdata/elf* directory. The defined values are:

0	SQL_TRIM_BOTH_	Remove blank space from beginning and end of string.
1	SQL_TRIM_BEGIN_	Remove blank space from beginning of string.

- | | | |
|---|----------------|--|
| 2 | SQL_TRIM_END_ | Remove blank space from end of string. |
| 3 | SQL_TRIM_NONE_ | Leave string as is. |

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

<- username@

Returns user name

Class DatasetClass get

Format value = this.username@

Description The get method username@ returns the user name used to connect to the database.

See [DatasetClass Methods](#) for more information.

<- value@

Returns a record column value

Class DatasetClass get

Format value = this.value@(column, record)

Arguments column A column name.

record A record number.

Description The get method value@ returns a record column value.

See [DatasetClass Methods](#) for more information.

<- vendor@

Returns database vendor

Class DatasetClass get

Format name = this.vendor@

Description The get method `vendor@` returns the vendor of the current database.

See [DatasetClass Methods](#) for more information.

<- group_by_info@

This method is obsolete, use the get method [group_by@](#).

<- having_info@

This method is obsolete, use the get method [having@](#).

-> having_info@

This method is obsolete, use the set method [having@](#).

-> group_by_info@

This method is obsolete, use the set method [group_by@](#).

-> auto_query@

Sets auto-query status

Class DatasetClass set

Format this.auto_query@(flag)

Arguments flag A Boolean value. TRUE enables auto-query, FALSE disables auto-query.

Description The set method auto_query@ sets the auto-query status. If auto-query is enabled, the database is re-queried on any change in the query information. If auto-query is not enabled, the database is only re-queried when the set method [requery@](#) is used.

See [DatasetClass Methods](#) for more information.

-> binary_size_limit@

Sets maximum data size

Class DatasetClass set

Format this.binary_size_limit@(limit)

Arguments limit The maximum size, in bytes, of binary data.

Description The set method binary_size_limit@ sets the maximum size of binary data allowed in the database. This value is database-dependent.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> bootstrap@

Initializes a data set

Class DatasetClass set

Format this.bootstrap@(vendor, table, database, host, server)

Arguments vendor The database vendor. The valid vendor names are:
Informix
Oracle
Sybase
table The name of the table in the database.

database	The name of the database to connect to. If a database name is not required, use NULL.
host	The host machine for the database.
server	The name of the database server.

Description The set method `bootstrap@` initializes a data set. If the arguments are not supplied to the method, then the data set information must be set with additional methods. If the arguments are supplied to the method, then the data set is initialized in a minimal state.

Use this method to create a data set programmatically and set the data set attributes. This method only initializes a data set with one table. If a data set with a join is required then you must set the data set information with the `tables@` or `join@` methods.

See [DatasetClass Methods](#) for more information.

-> `conditions@`

Sets query conditions

Class DatasetClass set

Format `this.conditions@(format arrayof sql_condition_info@ conditions)`

Arguments conditions An array of conditions.

Description The set method `conditions@` sets the query conditions of the current Data query to the passed array conditions. The structure `sql_condition_info@`, defined in the header file `dbase_.am` located in the `/install_dir/axdata/elf` directory, contains the following information:

format `sql_condition_info@`

table1,	The table in the database containing column1
column1,	The comparison column for the condition
compare,	(Not Used)
operate,	The comparison operator. The valid comparison operators are:
<	Less than
<=	Less than or equal
=	Equal (basic operator)
>=	Greater than or equal
>	Greater than
In	In (basic operator)
Between	Between (basic operator)
Like	Like (basic operator)
Is Null	Is Null (basic operator)

not_it,	Boolean value, if TRUE means use inverse of operator
value,	The constant value for a Value query
valueList,	A list of values for "In" or "Between" query condition
table2,	The table in the database containing column2
column2,	The column used for a Column query condition
or_mode,	Boolean value, where TRUE means OR, FALSE means AND. For use with multiple conditions, defines relation of this condition to previous condition
unused2,	(Not Used)
sqlstr,	A subselect string for a Subselect query
type,	Type of query: "Value", "Column", or "Subselect"
front_parens,	The number of left parentheses before the condition, for multiple conditions
rear_parens,	The number of right parentheses after the condition, for multiple conditions

See [DatasetClass Methods](#) for more information.

-> conditions_dlg@

Displays Conditions dialog box

Class DatasetClass set

Format this.conditions_dlg@

Description The set method conditions_dlg@ displays the Conditions dialog box used in the data set window. Use the Conditions dialog box to set or change the query conditions.

See [DatasetClass Methods](#) for more information.

-> connect@

Establishes database connection

Class DatasetClass set

Format this.connect@

Description The set method connect@ establishes a database connection. Use the set method define_database@ to define which database to connect to before using the connect@ method.

See [DatasetClass Methods](#) for more information.

-> `current_record_value@`

Sets a record column value

Class DatasetClass set

Format `this.current_record_value@(colName, value)`

Arguments `colName` The table column.
`value` The column value for the current record.

Description The set method `current_record_value@` sets the column value of the current record to the passed value.

For example, to set the column `last_name` in the current record to the value Jones you would call the method as:

```
this.current_record_value@("last_name", "Jones")
```

See [DatasetClass Methods](#) for more information.

-> `cursor_props@`

Sets cursor properties in the data set

Class DatasetClass set

Format `this.cursor_props@(format sql_cursor_properties@ props)`

Arguments `props` The structure `sql_cursor_properties@`, defined in the header file `dbase_.am` located in the `/install_dir/axdata/elf` directory, contains the following information:

`format sql_cursor_properties@`

`fetch_size,` The quantity of records (rows) to retrieve in a query of an Oracle® database server. This is an internal preference to potentially increase query speed, and does not have a visible effect on the number of records returned by the query. The default value is 50.

`trim_strings,` A value, as defined in the header file `dbase_.am`, located in the `install_dir/axdata/elf` directory. The defined values are:

SQL_TRIM_BOTH_ Remove blank space from beginning and end of string.

SQL_TRIM_BEGIN_ Remove blank space from beginning of string.

SQL_TRIM_END_ Remove blank space from end of string.

SQL_TRIM_NONE_ Leave string as is.

The default value is SQL_TRIM_BOTH_.

transpose_data, A Boolean value. TRUE transposes the columns and rows of the returned information. FALSE does not change the returned data. The default value is FALSE.

nums_as_nums, A Boolean value. TRUE signals Applixware Builder to handle numeric information in number format. FALSE signals Applixware Builder to handle numeric information as strings. The default value is FALSE.

binary_size_limit, The maximum size, in bytes, of binary data. This value is database-dependent.

timeout, The query timeout, in seconds. Use 0 for infinite limit, use NULL for default limit.

max_rows, The maximum quantity of records to retrieve. Pass NULL to get the default DBMS quantity.

dates_as_dates A Boolean value. TRUE signals Applixware Builder to handle date information in date format. FALSE signals Applixware Builder to handle date information as strings. The default value is FALSE.

Description The set method `cursor_props@` sets the cursor properties in the data set.

See [DatasetClass Methods](#) for more information.

-> datatype@

Sets data type

This method is not currently implemented.

See [DatasetClass Methods](#) for more information.

-> dates_as_dates@

Sets date handling

Class DatasetClass set

Format this.dates_as_dates@(flag)

Arguments flag A Boolean value. TRUE signals Applixware Builder to handle date information in date format. FALSE signals Applixware Builder to handle date information as strings.

Description The set method dates_as_dates@ sets the default date handling procedure for date information.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> date_format@

Sets date/time format

Class DatasetClass set

Format this.date_format@(format)

Arguments format The date time format, as defined in the header file datetim_.am located in the */install_dir/axdata/elf* directory.

Description The set method date_format@ sets the default date/time format of date and time values in the application. Include the datetim_.am header file in the object method source when you use defined date/time formats. Use the [format data for user](#) event to override the default format.

See [DatasetClass Methods](#) for more information.

-> decrement_cursor@

Moves position to previous record

Class DatasetClass set

Format this.decrement_cursor@

Description The set method decrement_cursor@ moves the database position back one record from the current record to the previous record.

NOTE: This is an obsolete method, use [decrement_position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> decrement_position@

Moves position to previous record

Class DatasetClass set

Format this.decrement_position@

Description The set method decrement_position@ moves the database position back one record from the current record to the previous record.

See [DatasetClass Methods](#) for more information.

-> define_database@

Sets the database connection information

Class DatasetClass set

Format this.define_database@(vendor, user, password, database,machine, server)

Arguments

vendor	The database vendor. The valid vendor names are: Informix Oracle Sybase
user	The user name used to connect to the database. If a user name is not required for the database, use NULL.
password	The password used to connect to the database. If a password is not required for the database, use NULL.
database	The name of the database to connect to. If a database name is not required, use NULL.
machine	The host machine for the database.
server	The name of the database server.

Description The set method `define_database@` sets the database connection information. After you define the database information, use the set method `connect@` to establish a connection to the database.

See [DatasetClass Methods](#) for more information.

-> `disconnect@`

Disconnects database

Class DatasetClass set

Format `this.disconnect@`

Description The set method `disconnect@` disconnects the current query from the database.

See [DatasetClass Methods](#) for more information.

-> `displayed_columns@`

Sets query column names

Class DatasetClass set

Format `this.displayed_columns@(columns)`

Arguments `columns` An array of column names.

Description The set method `displayed_columns@` sets the columns used for the data set query.

For example, to set the columns for the current data set you would call the method as:

```
var columns
columns[0] = "nameColumn"
columns[1] = "salaryColumn"
this.displayed_columns@(columns)
```

See [DatasetClass Methods](#) for more information.

-> display_binary_data@

Displays a BLOB in corresponding editor

Class DatasetClass set

Format this.display_binary_data@(blob)

Arguments blob A binary large object.

Description The set method display_binary_data@ displays a BLOB in the appropriate editor for the object. This method only works with the ODBC gateway. Use the SQLCommandClass get method [get_binary_data@](#) to retrieve BLOB data from the database.

See [DatasetClass Methods](#) for more information.

-> edit_apply_only@

Applies edits to database table without committing

Class DatasetClass set

Format this.edit_apply_only@

Description The set method edit_apply_only@ applies edits to database table without committing. If the database does not have transactions then the edits immediately update the database table, a commit is not required.

See [DatasetClass Methods](#) for more information.

-> edit_commit@

Updates database with edits, commits edits

Class DatasetClass set

Format this.edit_commit@(flag)

Arguments flag A Boolean variable, FALSE means requery the database after committing edits, TRUE means do not requery after committing edits.

Description The set method `edit_commit@` updates the database with all edits made against the current query, commits the edits, and closes the transaction. Once you commit edits, changes are made to the database. You cannot use `edit_rollback@` to undo the edits after they are committed.

See [DatasetClass Methods](#) for more information.

-> `edit_commit_only@`

Commits edits

Class DatasetClass set

Format `this.edit_commit_only@(flag)`

Arguments `flag` A Boolean variable, FALSE means requery the database after committing edits, TRUE means do not requery after committing edits.

Description The set method `edit_commit_only@` commits database edits and closes the transaction. The edits must be applied to the table before committing. Once you commit edits, changes are made to the database. You cannot use `edit_rollback@` to undo the edits after they are committed.

See [DatasetClass Methods](#) for more information.

-> `edit_delete@`

Deletes current record

Class DatasetClass set

Format `this.edit_delete@`

Description The set method `edit_delete@` deletes the current record from the database query. If you commit edits, the record is deleted from the database. Use `edit_rollback@` to undo a record deletion.

See [DatasetClass Methods](#) for more information.

-> edit_insert@

Inserts a database record

Class DatasetClass set

Format this.edit_insert@

Description The set method edit_insert@ inserts a database record in the current query. The record is inserted at the current position in the database.

See [DatasetClass Methods](#) for more information.

-> edit_rollback@

Cancels database edits

Class DatasetClass set

Format this.edit_rollback@(flag)

Arguments flag A Boolean variable, FALSE means requery the database after canceling edits, TRUE means do not requery after canceling edits.

Description The set method edit_rollback@ cancels all edits made against the current query. The method cannot undo edits that are committed to the database.

See [DatasetClass Methods](#) for more information.

-> edit_toggle_enabled@

Toggles edit mode

Class DatasetClass set

Format this.edit_toggle_enabled@

Description The set method edit_toggle_enabled@ toggles the edit mode. If editing is not enabled, the method enables editing. If editing is enabled, the method cancels all edits disables editing.

See [DatasetClass Methods](#) for more information.

-> edit_undo@

Removes edits from row and restores original state

Class DatasetClass set

Format this.edit_undo@([rowNum])

Arguments rowNum The row to unedit. This is an optional argument. If a row number is not passed the undo action is taken on the current row.

Description The set method edit_undo@ removes edits from a row and restores the row to its original state. Use this method to cancel edits in individual rows instead of using [edit_rollback@](#) to cancel all edits.

See [DatasetClass Methods](#) for more information.

-> exec_direct@

Executes an SQL statement

Class DatasetClass set

Format this.exec_direct@(sqlStmt)

Arguments sqlStmt The command being executed; sql_stmt can be a string or an array of strings.

Description Executes the passed statement directly. No explicit *prepare* statements or cursor calls need be used.

If sqlStmt is not a query statement, ELF SQL performs the following steps:

1. Prepares the statement.
2. Executes the statement.
3. Unprepares the statement.

If the statement is a query statement, the following sequence of operations occurs:

1. Prepares the statement.
2. Declares the cursor.

3. Opens the cursor.
4. Fetches all of the data.
5. Closes the cursor.
6. Unprepares the statement.

While it is very easy to invoke SQL statements using the `exec_direct@` method, it can greatly increase your overhead because of the added expense of repeatedly preparing the statement. Use the get method [exec_direct@](#) to execute an SQL statement that returns information.

See [DatasetClass Methods](#) for more information.

-> expressions@

Sets query expressions

Class DatasetClass set

Format this.expressions@(columns)

Arguments columns An array of expression columns.

Description The set method `expressions@` sets the expressions used in the database query. An expression is a query column created by using SQL functions with the columns from your database query. The functions include aggregate functions, such as sum (sum), average (avg), maximum (max), and minimum (min).

Expressions in the existing query must be included in the array of expressions.

See [DatasetClass Methods](#) for more information.

-> fetch_all_records@

Queries for all records

Class DatasetClass set

Format this.fetch_all_records@

Description The set method `fetch_all_records@` queries the database for all records matching the current query conditions.

See [DatasetClass Methods](#) for more information.

-> fetch_size@

Sets quantity of records to retrieve

Class DatasetClass set

Format this.fetch_size@(size)

Arguments size The quantity of records to retrieve.

Description The set method fetch_size@ sets the quantity of records (rows) to retrieve in a query of an Oracle® database server. This is an internal preference to potentially increase query speed, and does not have a visible effect on the number of records returned by the query. The default value is 50.

See [DatasetClass Methods](#) for more information.

-> group_by@

Sets the "group by" columns

Class DatasetClass set

Format this.group_by@(columns)

Arguments columns An array of table columns.

Description The set method group_by@ sets the "group by" columns for the query.

See [DatasetClass Methods](#) for more information.

-> having@

Sets the "having" conditions

Class DatasetClass set

Format this.having@(format arrayof sql_condition_info@ having)

Arguments having An array of structures containing "having" conditions.

Description The set method `having@` sets the query "having" conditions of the current query to the passed array of structures containing having conditions. The structure `sql_condition_info@`, defined in the header file `dbase_.am` located in the `/install_dir/axdata/elf` directory, contains the following information:

```
format sql_condition_info@
table1,           The table in the database containing column1
column1,         The comparison column for the condition
compare,         (Not Used)
operate,         The comparison operator. The valid comparison operators are:
<               Less than
<=              Less than or equal
=               Equal (basic operator)
>=              Greater than or equal
>               Greater than
In               In (basic operator)
Between         Between (basic operator)
Like            Like (basic operator)
Is Null         Is Null (basic operator)
not_it,         Boolean value, if TRUE means use inverse of operator
value,          The constant value for a Value query
valueList,      A list of values for "In" or "Between" query condition
table2,         The table in the database containing column2
column2,        The column used for a Column query condition
or_mode,        Boolean value, where TRUE means OR, FALSE means AND. For
                use with multiple conditions, defines relation of this condition to pre-
                vious condition
unused2,        (Not Used)
sqlstr,         A subselect string for a Subselect query
type,           Type of query: "Value", "Column", or "Subselect"
front_parens,   The number of left parentheses before the condition, for multiple
                conditions
rear_parens,    The number of right parentheses after the condition, for multiple
                conditions
```

See [DatasetClass Methods](#) See [DatasetClass Methods](#) for more information.

-> `increment_cursor@`

Moves position to next record

Class DatasetClass set

Format this.increment_cursor@

Description The set method increment_cursor@ moves the database position forward one record from the current record to the next record.

NOTE: This is an obsolete method, use [increment_position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> increment_position@

Moves position to next record

Class DatasetClass set

Format this.increment_position@

Description The set method increment_position@ moves the database position forward one record from the current record to the next record.

See [DatasetClass Methods](#) for more information.

-> is_editable_by_default@

Sets initial editing status

Class DatasetClass set

Format this.is_editable_by_default@(flag)

Arguments flag A Boolean value. TRUE allows editing by default. FALSE does not allow editing by default.

Description The set method is_editable_by_default@ sets the initial editing status of a data set. Use this method in the initialize_event of the dialog box to allow immediate editing in the dialog box. For example, the following initialize_event enables editing:

```
set initialize_event
    var object dataset
    dataset = this.data_set@
    dataset.is_editable_by_default@ = TRUE
endset
```

See [DatasetClass Methods](#) for more information.

-> [is_editing_enabled@](#)

Sets editing status

Class DatasetClass set

Format this.is_editing_enabled@(flag)

Arguments flag A Boolean value. TRUE enables editing. FALSE disables editing.

Description The set method [is_editing_enabled@](#) sets the database editing status.

See [DatasetClass Methods](#) for more information.

-> [is_transactionless@](#)

Obsolete

This method is obsolete, see [join@](#) for more information.

-> [join@](#)

Sets join information

Class DatasetClass set

Format this.join@(format arrayof sql_join_info@ joinInfo)

Arguments joinInfo Information in [sql_join_info@](#) format. The format is defined in the `dbase_.am` header file, located in the `install_dir/axdata/elf` directory. The format contains the following attributes:

format [sql_join_info@](#)

table1,	Name of the first table
table2,	Name of the second table
column1,	Name of the join column in table1
column2,	Name of the join column in table2
outer1,	Boolean value where TRUE means a right outer join
outer2,	Boolean value where TRUE means a left outer join
operator,	One of the following comparison operators:

= equal
!= not equal
< less than
<= less than or equal
> greater than
>= greater than or equal

Description The set method `join@` sets join information for the current query. If a table join is defined, then the array of join information contains at least two elements. The first element is a join of the first table to itself. The second element is a join of the first table on the second table.

See [DatasetClass Methods](#) for more information.

-> `join_info@`

Sets join information

This method is obsolete, use the `join@` method to set table join information.

-> `load_query_from_file@`

Loads an Applixware Data query

Class DatasetClass set

Format `this.load_query_from_file@(file)`

Arguments file An Applixware Data file, with full path name and .aq file extension, or information in `sql_state@` format. The format is defined in the `dbase_.am` header file, located in the `install_dir/axdata/elf` directory.

Description The set method `load_query_from_file@` loads an Applixware Data query as the current database query.

See [DatasetClass Methods](#) for more information.

-> `lock_on_edit@`

Sets row locking operation

Class DatasetClass set

Format this.lock_on_edit@(flag)

Arguments flag A Boolean value. FALSE does not lock on edit, TRUE locks on edit. The default is TRUE.

Description The set method lock_on_edit@ sets the row locking operation. Use the method to set row locking when you edit any value in a row.

See [DatasetClass Methods](#) for more information.

-> login@

Sets database login info

Class DatasetClass set

Format this.login@(user, pwd, flag)

Arguments user The database user name.
pwd The user password.
flag The initialization state, as a Boolean value. TRUE means the user name and password are initialized. The initialization state helps you determine if a user name or password were deliberately set to NULL, or if they are not set.

Description The set method login@ sets the user name and password in the current data set for database login. This overrides the [login_global@](#) login settings.

See [DatasetClass Methods](#) for more information.

-> login_dlg@

Displays login dialog box

Class DatasetClass set

Format this.login_dlg@

Description The set method login_dlg@ displays the Login Identification dialog box. Use the dialog box as a consistent user interface for setting the user name and password for database login.

See [DatasetClass Methods](#) for more information.

-> login_global@

Sets database login info

Class DatasetClass set

Format this.login_global@(user, pwd)

Arguments user The database user name.
pwd The user password.

Description The set method login_global@ sets the user name and password for all data sets for database login. All data sets use this information by default, unless login information is set for the specific data set with the [login@](#) method

See [DatasetClass Methods](#) for more information.

-> max_records@

Sets maximum quantity of records

Class DatasetClass set

Format this.max_records@(limit)

Arguments limit The maximum quantity of records to retrieve.

Description The set method max_records@ sets the maximum quantity of records retrieved by a query. Limiting the number of records you retrieve may reduce the time spent executing a query. The default is 5,000.

See [DatasetClass Methods](#) for more information.

-> no_duplicates@

Sets "allow duplicates" status

Class DatasetClass set

Format this.no_duplicates@(flag)

Arguments flag A Boolean value. FALSE allows duplicate rows in the query, TRUE does not allow duplicate rows in the query.

Description The set method no_duplicates@ sets the "allow duplicates" status of the query. See [DatasetClass Methods](#) for more information.

-> numbers_as_numbers@

Sets number handling

Class DatasetClass set

Format this.numbers_as_numbers@(flag)

Arguments flag A Boolean value. TRUE signals Applixware Builder to handle numeric information in number format. FALSE signals Applixware Builder to handle numeric information as strings.

Description The set method numbers_as_numbers@ sets the default handling procedure for numeric information. Numbers are converted to strings by default.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> order_by@

Sets sort order

Class DatasetClass set

Format this.order_by@(format arrayof sql_sort_info@ sort)

Arguments sort The sort information.

Description The set method order_by@ sets the sort order information in the data set query. The structure sql_sort_info@, defined in the header file dbase_.am located in the */install_dir/axdata/elf* directory, contains the following information:

format sql_sort_info@
column_name, The name of the column determining the sort

descending	A Boolean value, where TRUE means descending sort, and FALSE means ascending sort.
descending	Boolean value where TRUE means descending sort, and FALSE means ascending sor

See [DatasetClass Methods](#) for more information.

-> position@

Sets current position and updates dialog box

Class DatasetClass set

Format this.position@(value)

Arguments value A numeric record value. Records are 0-based.

Description The set method position@ sets the database position to the passed value and updates related dialog box widgets containing data set information.

For example, to set the position to the next record you would call the method as:

```
var value
value = this.position@    'Get current record number
value = value +1        'Set to next record
this.position@(value)
```

You cannot use position@ with the set method exec_direct@, exec_direct@ does not open a cursor to the table.

See [DatasetClass Methods](#) for more information.

-> query_changed@

Signals a change in query

Class DatasetClass set

Format this.query_changed@

Description The set method query_changed@ signals a change in the current query. If auto_query@ is set to TRUE, the database is automatically requeried with the new query conditions.

See [DatasetClass Methods](#) for more information.

-> [requery@](#)

Queries database for data set update

Class DatasetClass set

Format this.requery@

Description The set method [requery@](#) queries the database to update the data set and related dialog box widgets containing data set information. Use this method when [auto_query@](#) is set to FALSE.

See [DatasetClass Methods](#) for more information.

-> [set_cursor@](#)

Sets current position and updates dialog box

Class DatasetClass set

Format this.set_cursor@(value)

Arguments value A numeric record value. Records are 0-based.

Description The set method [set_cursor@](#) sets the database position to the passed value and updates related dialog box widgets containing data set information.

For example, to set the position to the next record you would call the method as:

```
var value
value = this.get_cursor@ 'Get current record number
value = value +1 'Set to next record
this.set_cursor@(value)
```

NOTE: This is an obsolete method, use [position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> set_position@

Sets current position and updates dialog box

Class DatasetClass set

Format this.set_position@(value)

Arguments value A numeric record value. Records are 0-based.

Description The set method set_position@ sets the database position to the passed value and updates related dialog box widgets containing data set information.

For example, to set the position to the next record you would call the method as:

```
var value
value = this.position@    'Get current record number
value = value +1        'Set to next record
this.position@(value)
```

NOTE: This is an obsolete method, use [position@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> sort_dlg@

Displays Sort dialog box

Class DatasetClass set

Format this.sort_dlg@

Description The set method sort_dlg@ displays the Sort dialog box used in the data set window. Use the Sort dialog box to set or change the sort order of information retrieved by a query.

See [DatasetClass Methods](#) for more information.

-> **sort_info@**

Sets sort information

Class DatasetClass set

Format this.sort_info@(format arrayof sql_sort_info@ sort)

Arguments sort The sort information.

Description The set method `sort_info@` sets the sort information for the current data query. The structure `sql_sort_info@`, defined in the header file `dbase_.am` located in the `/install_dir/axdata/elf` directory, contains the following information:

format `sql_sort_info@`

`column_name`, The name of the column determining the sort

`descending` Boolean value where TRUE means descending sort, and FALSE means ascending sort

NOTE: This is an obsolete method, use [order_by@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> **source@**

Sets SQL source code

Class DatasetClass set

Format this.source@(strings)

Arguments strings An array of strings containing SQL statements.

Description The set method `source@` sets the SQL source code with a passed array of strings.

See [DatasetClass Methods](#) for more information.

-> **status_message@**

Sets status message

Class DatasetClass set

Format this.status_message@(string)

Arguments string The message string.

Description The set method status_message@ sets the status message.

See [DatasetClass Methods](#) for more information.

-> tables@

Sets table information

Class DatasetClass set

Format this.tables@(format arrayof sql_table_info@ tables, format arrayof sql_join_info@ joins, flag)

Arguments

tables	Table information in sql_table_info@ format. The format is defined in the dbase_.am header file, located in the <i>install_dir/axdata/elf</i> directory.
joins	Join information in sql_join_info@ format. The format is defined in the dbase_.am header file, located in the <i>install_dir/axdata/elf</i> directory. If a table join is defined, then the array of join information contains at least two elements. The first element is a join of the first table to itself. The second element is a join of the first table on the second table.
flag	A Boolean value. TRUE means reset the current query info, FALSE means don't reset the current query information. The default value is TRUE.

Description The set method tables@ sets the table information for the current query.

See [DatasetClass Methods](#) for more information.

-> temp_status_message@

Displays a temporary status message

Class DatasetClass set

Format this.temp_status_message@(string, time)

Arguments

string	The message string.
time	The duration of the message, in seconds.

Description The set method `temp_status_message@` displays a temporary status message. Subsequent calls to `status_message@` will override the temporary status message, even if the time for the temporary message has not expired.

See [DatasetClass Methods](#) for more information.

-> `timeout@`

Sets query timeout limit

Class DatasetClass set

Format `this.timeout@(time)`

Arguments time The query timeout, in seconds.

Description The set method `timeout@` sets the query timeout limit. The method limits the duration of query. Use the method to avoid time consuming queries, or to cancel queries that would continue indefinitely.

NOTE: This is an obsolete method, use `cursor_props@` to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> `transpose_data@`

Sets data transpose status

Class DatasetClass set

Format `this.transpose_data@(flag)`

Arguments flag A Boolean value. TRUE transposes the columns and rows of the returned information. FALSE does not change the returned data.

Description The set method `transpose_data@` sets the data transpose status. For example, if the method is set to TRUE the information originally retrieved as `data[2,1]` becomes `data[1,2]`.

NOTE: This is an obsolete method, use `cursor_props@` to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> trim_strings@

Determines blank space handling in strings

Class DatasetClass set

Format this.trim_strings@(value)

Arguments value A value, as defined in the header file `dbase_.am`, located in the `install_dir/axdata/elf` directory. The defined values are:

SQL_TRIM_BOTH_	Remove blank space from beginning and end of string.
SQL_TRIM_BEGIN_	Remove blank space from beginning of string.
SQL_TRIM_END_	Remove blank space from end of string.
SQL_TRIM_NONE_	Leave string as is.

Description The set method `trim_strings@` determines how blank spaces are handled with returned strings.

NOTE: This is an obsolete method, use [cursor_props@](#) to guarantee compatibility with future releases.

See [DatasetClass Methods](#) for more information.

-> value@

Sets a field value in a record

Class DatasetClass set

Format this.value@(column, recordNum, value)

Arguments column A column name.
recordNum A numeric record value. Records are 0-based. Pass NULL for the current record.
value A new field value.

Description The set method `value@` sets a field value in a record to the passed value. If editing is not enabled, or the column is not in the current data set, an error is thrown.

See [DatasetClass Methods](#) for more information.

<- error_event

Called for object errors

Class DatasetClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Appixware Builder.

Description The error_event is called by Appixware Builder for posting the object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [DatasetClass Methods](#) for more information.

<- format_data_for_dbms

Called after editing a field

Class DatasetClass event

Format retVal = this.format_data_for_dbms(index, value)

Arguments index The column index.
value The edited value of the field.

Description The format_data_for_dbms event is called by Appixware Builder to format a value in the current row before sending it to the database. The method should return a value if you format the value in the event. The method should return NULL if you want Appixware Builder to format the value with the default event handler.

See [DatasetClass Methods](#) for more information.

<- format_data_for_user

Called when displaying a field

Class DatasetClass event

Format this.format_data_for_user(index, value)

Arguments index The column index.
value The current value in the row.

Description The format_data_for_user event is called by Applixware Builder to format a value in the current row before sending it to the application dialog box controls. The method should return a value if you format the value in the event. The method should return NULL if you want Applixware Builder to format the value with the default event handler.

See [DatasetClass Methods](#) for more information.

-> initialize_event

Called when starting an application

Class DatasetClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder to initialize DatasetClass objects. This event is only called once, after the application's initialize_event. This is a user-defined event, place all actions to initialize the object attributes and system variables in the event definition.

See [DatasetClass Methods](#) for more information.

-> terminate_event

Called before closing an application

Class DatasetClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing an application. This event is only called once, before the application's `terminate_event`. This is a user-defined event, place all actions to free memory, reset attributes, and remove system variables in the event definition.

See [DatasetClass Methods](#) for more information.

-> `time_out_event`

Called on object timer time-out

Class `DatasetClass event`

Format `this.time_out_event`

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [DatasetClass Methods](#) for more information.

<- `accept_pokes@`

Returns messages dialog box accepts

Class `DialogBoxClass get`

Format `pokeArray = this.accept_pokes@`

Description The `get` method `accept_pokes@` returns an array of message codes accepted by the dialog box. The acceptable poke codes are set with the `set` method `accept_pokes@` and passed to the `poke_event`. You can define your own poke events, or use the poke codes defined in the `/installdir/axdata/elf/dlgpoke_.am` header file.

For example, to get the accepted poke codes use the following:

```
include "dlgpoke_.am"  
var pokeArray  
pokeArray = this.accept_pokes@
```

<- arg_var@

Returns dialog box arguments

Class DialogBoxClass get

Format argument = this.arg_var@(index)

Arguments index The index in the array of arguments.

Description The get method arg_var@ returns an argument from the array of arguments passed to the dialog box.

For example, to get argument 0 use the following:

```
var argument  
argument = this.arg_var@(0)
```

<- data_set@

Returns data set associated with dialog box

Class DialogBoxClass get

Format obj = this.data_set@

Description The get method data_set@ returns the data set associated with the dialog box. If a data set is not associated with the dialog box, the method returns NULL.

For example, to get the data set associated with the dialog box use the following:

```
var object obj  
obj = this.data_set@
```

<- *expressline_status@*

Returns *ExpressLine* status in dialog box

Class DialogBoxClass get

Format val = this.*expressline_status@*

Description The get method *expressline_status@* returns a value for the *ExpressLine* status in the dialog box. The values, as defined in the header file *menubar_.am*, located in the *install_dir/axdata/elf* directory are:

NULL	The dialog box does not have a menu bar.
1 MENUSTAT#DIMMED	The dialog box has a menu bar, but no <i>ExpressLine</i> .
2 MENUSTAT#TOGGLE_ON	The dialog box has a menu bar and <i>ExpressLine</i> , but <i>ExpressLine</i> display is off.
3 MENUSTAT#TOGGLE_OFF	The dialog box has a menu bar and <i>ExpressLine</i> , <i>ExpressLine</i> display is on.

<- *height@*

Returns dialog box height

Class DialogBoxClass get

Format pixels = this.*height@*

Description The get method *height@* returns the height, in pixels, of the dialog box.

For example, to get the height of the dialog box and set it to a value use the following:

```
var pixels
pixels = this.height@           'get method
IF pixels < 200
    this.height@(200)         'set method
```

<- help_topic@

Returns dialog box help hypertarget

Class DialogBoxClass get

Format target = this.help_topic@

Description The get method help_topic@ returns the dialog box help hypertarget. The hypertarget is only available for help written in the Applixware Help system. Help is unavailable if the application is running against elfrt. Use the [post_help@](#) method to display a help topic in the Applixware Help system.

For example, to verify and set a dialog box help hypertarget use the following:

```
var target
target = this.help_topic@      'get method
IF target <> "dialog_help"
    this.help_topic@("dialog_help") 'set method
```

<- iconize@

Returns TRUE if the dialog box is iconized

Class DialogBoxClass get

Format flag = this.iconize()

Description Returns TRUE if the dialog box is iconized. Otherwise, this method returns FALSE.

<- icon_name@

Returns dialog box icon name

Class DialogBoxClass get

Format icon = this.icon_name@

Description The get method icon_name@ returns the dialog box set-aside icon number. The returned number is used with the Icon Size preference setting to reference the .im bitmap file.

For example, if the method returns 121 and the Icon Size preference is set to 50x50, the name of the required bitmap file is 121-50x50.im. The possible icon sizes are 16, 32, 50, and 64. You may require bitmap files of an image in various sizes to have a set-aside icon available for each preference setting.

<- icon_title@

Returns dialog box icon title

Class DialogBoxClass get

Format title = this.icon_title@

Description The get method icon_title@ returns the dialog box set-aside icon title. You can change the title of the set-aside icon to reflect changes in your application.

<- is_cancel_destroy@

Returns destroy cancelling status of dialog box

Class DialogBoxClass get

Format flag = this.is_cancel_destroy@

Description The get method is_cancel_destroy@ returns a Boolean value for the destroy cancelling status of the dialog box. The method returns TRUE if the destroy cancelling status is enabled, otherwise it returns FALSE. Use the set method [is_cancel_destroy@](#) in the [destroy event](#) to cancel the destruction of dialog box.

<- is_expand@

Returns dialog box auto-scale status

Class DialogBoxClass get

Format flag = this.is_expand@

Description The get method is_expand@ returns a Boolean value for the dialog box auto-scale status. The method returns TRUE if the dialog box auto-scaling is enabled, otherwise it returns FALSE. Auto-scaling or expansion allows dialog boxes created in one font mode and displayed in a new font mode to scale or expand to fit the new font mode. This

method takes effect when changing the Use Larger Font Size for Menus ([axFontsBig-Menus](#)) preference in Screen Display Preferences.

See [DialogBoxClass Methods](#) for more information.

<- is_open@

Returns dialog box open status

Class DialogBoxClass get

Format flag = this.is_open@

Description The get method is_open@ returns a Boolean value for the dialog box open status. The method returns TRUE if the dialog box is open, otherwise it returns FALSE.

For example, to get the dialog box open status use the following:

```
flag = this.is_open@
```

See [DialogBoxClass Methods](#) for more information.

<- keyfocus_widget@

Returns control with keyboard focus

Class DialogBoxClass get

Format object ctl = this.keyfocus_widget@

Description The get method keyfocus_widget@ returns a reference to the control that has the keyboard focus. The method returns NULL if the dialog box is not open, or if no control in the dialog box has focus.

See [DialogBoxClass Methods](#) for more information.

<- menu_bar@

Returns dialog box menu bar

Class DialogBoxClass get

Format menu = this.menu_bar@

Description The get method `menu_bar@` returns the menu bar associated with the dialog box. If a menu bar is not associated with the dialog box, the method returns NULL.

For example, to get the dialog box menu bar use the following:

```
var object menu
menu = this.menu_bar@
```

See [DialogBoxClass Methods](#) for more information.

`<- open@`

Displays dialog box, returns close value

Class DialogBoxClass get

Format `value = this.open@([arg1[,arg2[,arg3[,arg4[,arg5]]]])`

Arguments `arg1..arg5` Optional user values required by the dialog box.

Description The get method `open@` opens a dialog box or promotes a dialog box if it is already open. The method returns a user value if the dialog box is a response dialog box.

The open action for each dialog box type is as follows:

Normal Opens/promotes the dialog box.

Response Opens the dialog box and pends for a returned user value.

Multiple Opens the dialog box and returns new dialog box object instance handle.

Main Opened by run-time environment when application is executed.

For example, to open a response dialog box with the argument Orange use the following:

```
var value
value = this.open@("Orange")
```

See [DialogBoxClass Methods](#) for more information.

`<- return_value@`

Returns dialog box return value

Class DialogBoxClass get

Format value = this.return_value@

Description The get method return_value@ returns a user value if the dialog box returns a value. Use the method to test the return value set for a response dialog box with the set method [return_value@](#). Use the set method [type@](#) to define a response dialog box using the DBOX_TYPE#RESPONSE_ defined type. Dialog box types are defined in the header file builder_.am, located in the *install_dir/axdata/elf* directory.

For example, to get the dialog box return value use the following:

```
var value
value = this.return_value@
```

See [DialogBoxClass Methods](#) for more information.

<- title@

Returns a dialog box title

Class DialogBoxClass get

Format title = this.title@

Description The get method title@ returns the dialog box title as a string. You can use the method with the set method title@ to verify and set the dialog box title.

For example, to verify a dialog box title use the following:

```
var oldTitle
oldTitle = this.title@           'get method
if oldTitle <> "Orginal Dialog Box Title"
this.title@("Orginal Dialog Box Title") 'set method
```

See [DialogBoxClass Methods](#) for more information.

<- type@

Returns dialog box type

Class DialogBoxClass get

Format type = this.type@

Description The get method `type@` returns the dialog box type. The dialog box types are defined in the header file `builder_.am`, located in the `install_dir/axdata/elf` directory. The valid dialog box types are:

- 0 `DBOX_TYPE#NORMAL_` Normal. The standard dialog box type, only 1 instance of the dialog box is allowed at one time
- 1 `DBOX_TYPE#MAIN_` Main. The dialog box that appears when you run a Builder application. You can designate only one Main dialog box.
- 2 `DBOX_TYPE#RESPONSE_` Response. The application action is suspended, waiting for information from the dialog box. Application action resumes when the dialog box is dismissed.
- 3 `DBOX_TYPE#MULTIPLE_` Multiple. The dialog box can have multiple instances at the same time.

For example, to get the dialog box type use the following:

```
var type
type = this.type@
```

See [DialogBoxClass Methods](#) for more information.

`<- width@`

Returns dialog box width

Class DialogBoxClass get

Format `pixels = this.width@`

Description The get method `width@` returns the width, in pixels, of the dialog box. You can use this method with the get method `width@` to verify and set a dialog box width.

For example, to set a dialog box width use the following:

```
var pixels
pixels = this.width@      'get method
IF pixels < 200
    this.width@(200)     'set method
```

See [DialogBoxClass Methods](#) for more information.

<- x_pos@

Returns dialog box x-axis position

Class DialogBoxClass get

Format pos = this.x_pos@

Description The get method x_pos@ returns the dialog box x-axis position, in pixels, of the top left corner of the dialog box on the screen. The maximum values for x_pos@ and y_pos@ will depend on the dimensions of your screen. The closest dialog box position to the top left of a screen is position (1,0) or position (0,1).

See [DialogBoxClass Methods](#) for more information.

<- y_pos@

Sets dialog box y-axis position

Class DialogBoxClass get

Format pos = this.y_pos@

Description The get method y_pos@ returns the dialog box y-axis position, in pixels, of the top left corner of the dialog box on the screen. The maximum values for x_pos@ and y_pos@ will depend on the dimensions of your screen. The closest dialog box position to the top left of a screen is position (1,0) or position (0,1).

See [DialogBoxClass Methods](#) for more information.

-> accept_pokes@

Sets messages dialog box accepts

Class DialogBoxClass set

Format this.accept_pokes@(pokeArray)

Arguments pokeArray An array of poke message codes.

Description The set method `accept_pokes@` sets the poke message codes accepted by the dialog box. You can define your own poke events, or use the poke codes defined in the `/install/axdata/elf/dlgpoke_.am` header file.

For example, to set the accepted poke codes use the following:

```
include "dlgpoke_.am"
this.accept_pokes@ = {POKE_EXIT, POKE_DIALOG_RESIZED}
```

-> `add_child@`

Adds an object as a child

Class DialogBoxClass set

Format `this.add_child@(object)`

Arguments object An object created with the `obj_create@` macro.

Description The set method `add_child@` adds an object to the current object's list of children.

For example, to add an object `buttonExample` to the current object use the following:

```
this.add_child@(buttonExample)
```

-> `arg_var@`

Sets an argument value

Class DialogBoxClass set

Format `this.arg_var@(index, value)`

Arguments index The index in the array of arguments.
value The new value for the argument.

Description The set method `arg_var@` changes an argument passed to the dialog box by the [open@](#) method. Use this method with the get method `arg_var@` to check and change arguments.

For example, to set argument 1 to Blue use the followings:

```
var argument
```

```
argument = this.arg_var@(1)    ' get method
IF argument <> "Blue"
    this.arg_var@(1, "Blue") ' set method
```

-> arg_vars@

Sets argument values

Class DialogBoxClass set

Format this.arg_vars@(valArray)

Arguments valArray An array of argument values.

Description The set method arg_vars@ changes the arguments passed to the dialog box by the [open@](#) method. Use this method with the get method arg_vars@ to check and change arguments.

-> close@

Closes a dialog box

Class DialogBoxClass set

Format this.close@

Description The set method close@ closes a dialog box.
For example, to close a dialog box use the followings:

```
this.close@
```

-> close_with_return@

Closes a response dialog box with a return value

Class DialogBoxClass set

Format this.close_with_return@(value)

Arguments value A user-defined value returned by the response dialog box.

Description The set method `close_with_return@` closes a response dialog box and sets the value returned by the response dialog box.

For example, to close a response dialog box with the value `FALSE` use the followings:

```
this.close_with_return@(FALSE)
```

-> `copy@`

Copies text from an edit box or entry field to the clipboard

Class DialogBoxClass set

Format `this.copy()`

Description Copies highlighted text from an edit box or entry field to the clipboard. The widget containing the highlighted text must have keyboard focus in order for the copy to be successful. Use the DialogBoxClass method `keyfocus_widget@` to set the focus to the dialog box control that you wish to copy from.

-> `customize_menu_bar_dlg@`

Saves customized menu bar in a file

Class DialogBoxClass set

Format `this.customize_menu_bar_dlg@(file)`

Arguments file The file name of the customized menu bar. The file is saved in the `/user/axhome` directory.

Description The set method `customize_menu_bar_dlg@` allows you to customize the dialog box menu bar of a distributed application and save it to a file. Use the method with `load_custom_menu_bar@` to create custom menu bars and load them in a dialog box. You are unable to add methods to the objects in a distributed application. You can only add menu items to a customized menu bar that call macros.

-> `cut@`

Cuts text from an edit box or entry field to the clipboard

Class DialogBoxClass set

Format this.cut()

Description Cuts highlighted text from an edit box or entry field and places it on the clipboard. The widget containing the highlighted text must have keyboard focus in order for the cut to be successful. Use the DialogBoxClass method [keyfocus_widget@](#) to set the focus to the dialog box control that you wish to cut from.

-> data_set@

Sets data set associated with dialog box

Class DialogBoxClass set

Format this.data_set@(object data)

Arguments data A data set in the application.

Description The set method data_set@ sets the data set associated with the dialog box. If a data set does not exist, the data set associated with the dialog box is set to NULL.

-> delete_child@

Removes a child object from dialog box

Class DialogBoxClass set

Format this.delete_child@(objectName)

Arguments objectName The child object to delete.

Description The set method delete_child@ removes the passed child object from the current dialog box. Use the get method child@ to get a pointer to the object, then pass the returned object to delete_child@.

For example, to remove the child object exampleButton use the following:

```
var object childObject
childObject = this.child@("exampleButton")
this.delete_child@(childObject)
```

-> disable@

Disables a dialog box

Class DialogBoxClass set

Format this.disable@

Description The set method disable@ disables a dialog box. All widgets in the dialog box are grayed and become inactive. Use this method to make dialog box actions unavailable without closing the dialog box.

For example, to disable a dialog box use the following:

```
    this.disable@
```

-> display@

Displays a dialog box

Class DialogBoxClass set

Format this.display@

Description The set method display@ displays a dialog box..

-> enable@

Enables a dialog box

Class DialogBoxClass set

Format this.enable@

Description The set method enable@ enables a dialog box. All widgets in the dialog box are ungrayed and initialized.

For example, to enable a dialog box use the following:

```
    this.enable@
```

-> execute_and_dismiss@

Executes a dialog box action, then dismisses

Class DialogBoxClass set

Format this.execute_and_dismiss@

Description The set method execute_and_dismiss@ executes the dialog box actions defined in execute_event, then the actions in dismiss_event before closing the dialog box.

-> height@

Sets dialog box height

Class DialogBoxClass set

Format this.height@(newHeight)

Arguments newHeight The new dialog box height, in pixels.

Description The set method height@ sets the new dialog box height to the passed value. You can use this method with the get method height@ to set a dialog box height.

For example, to set a dialog box height use the followings:

```
var pixels
pixels = this.height@           'get method
IF pixels < 200
    this.height@(200)         'set method
```

-> help_topic@

Sets dialog box help hypertext

Class DialogBoxClass set

Format this.help_topic@(target)

Arguments target The help hypertext.

Description The set method `help_topic@` sets the dialog box help hypertarget. The hypertarget is only available for help written in the Applixware Help system. Help is unavailable if the application is running against elfrt.

For example, to verify and set a dialog box help hypertarget use the following:

```
var target
target = this.help_topic@      'get method
IF target <> "dialog_help"
    this.help_topic@("dialog_help") 'set method
```

-> **iconize@**

Minimizes a dialog box

Class DialogBoxClass set

Format DialogBoxClass->iconize@(flag)

Arguments flag A Boolean. If TRUE, the dialog box is minimized. If FALSE, the dialog box is maximized.

Description Minimizes the dialog box, and sets it aside as an icon. If the FLAG argument is false, the icon is maximized. (This is equivalent to double-clicking the icon in an X-windows environment.)

-> **icon_name@**

Sets dialog box icon name

Class DialogBoxClass set

Format this.icon_name@(icon)

Arguments icon The icon number.

Description The set method `icon_name@` sets the dialog box set-aside icon number. The passed number is used with the Icon Size preference setting to reference the .im bitmap file.

For example, if use the method as follows

```
this.icon_name@(121)
```

and the Icon Size preference is set to 50x50, the name of the required bitmap file is 121-50x50.im. The possible icon sizes are 16, 32, 50, and 64. You may require bitmap files of an image in various sizes to have a set-aside icon available for each preference setting.

-> icon_title@

Sets dialog box icon title

Class DialogBoxClass set

Format this.icon_title@(title)

Arguments icon The icon title.

Description The set method icon_name@ sets the dialog box set-aside icon title. You can change the title of the set-aside icon to reflect changes in your application.

-> is_cancel_destroy@

Sets destroy cancelling status of dialog box

Class DialogBoxClass set

Format this.is_cancel_destroy@(flag)

Arguments flag A Boolean value, TRUE cancels the destruction of a dialog box, FALSE enables the destruction of a dialog box. The default value is FALSE.

Description The set method is_cancel_destroy@ sets the destroy cancelling status of the dialog box. Use this method in the [destroy_event](#) to cancel the destruction of dialog box, it has no effect outside of the destroy_event.

-> is_expand@

Sets dialog box auto-scale status

Class DialogBoxClass set

Format this.is_expand@(flag)

Arguments flag A Boolean value, TRUE enables auto-scaling, FALSE disables auto-scaling.

Description The set method `is_expand@` sets the dialog box auto-scale status. Auto-scaling or expansion allows dialog boxes created in one font mode and displayed in a new font mode to scale or expand to fit the new font mode. This method takes effect when changing the Use Larger Font Size for Menus ([axFontsBigMenus](#)) preference in Screen Display Preferences.

-> **keyfocus_widget@**

Sets the control receiving keyboard focus

Class DialogBoxClass set

Format `this.keyfocus_widget@(object ctl)`

Description The set method `keyfocus_widget@` sets the control that receives keyboard focus in the dialog box. Assign keyboard focus to a control to allow keyboard actions without having to choose the control with the mouse.

-> **load_custom_menu_bar@**

Loads customized menu bar into a dialog box

Class DialogBoxClass set

Format `this.load_custom_menu_bar@(file)`

Arguments file The full path name name of the customized menu bar file.

Description The set method `load_custom_menu_bar@` allows you to load a customized menu bar into the dialog box of a distributed application. Use the method with [customize menu bar dlg@](#) to create custom menu bars and load them in a dialog box.

Use this method in the dialog box `initialize_event` to load the customized menu bar.

-> menu_bar@

Sets dialog box menu bar

Class DialogBoxClass set

Format this.menu_bar@(object menu)

Arguments menu A MenuBarClass object defining a menu.

Description The set method menu_bar@ sets the menu bar associated with the dialog box.

-> open@

Opens a dialog box

Class DialogBoxClass set

Format this.open@([arg1[,arg2[,arg3[,arg4[,arg5]]]])

Arguments arg1..arg5 Optional user values required by the dialog box.

Description The set method open opens a dialog box.

The open action for each dialog box type is as follows:

Normal Opens/promotes the dialog box.

Response Opens the dialog box and pends for a returned user value.

Multiple Opens the dialog box and returns new dialog box object instance handle.

Main Opened by run-time environment when application is executed.

For example, to open a dialog box with the argument Orange use the following:

this.open("Orange")

-> paste@

Pastes text to an edit box or entry field

Format this.paste()

Description Pastes text that was previously copied to the clipboard to an edit box or entry field in the dialog box. The edit box or entry field must have keyboard focus in order for the paste to

be successful. Use the DialogBoxClass method [keyfocus_widget@](#) to set the focus to the dialog box control before you paste.

-> [post_help@](#)

Calls dialog box help

Class DialogBoxClass set

Format this.post_help@

Description The set method [post_help@](#) calls dialog box help. The method passes the hypertext defined with the set method [help_topic@](#) to the Applixware Help system. The hypertext is only available for help written in the Applixware Help system. Help is unavailable if the application is running against elfrt.

-> [return_value@](#)

Sets dialog box return value

Class DialogBoxClass set

Format this.return_value@(value)

Arguments value A user value returned by the dialog box.

Description The set method [return_value@](#) sets a user value returned by a response dialog box. Use the set method [type@](#) to define a response dialog box using the `DBOX_TYPE#RESPONSE_` defined type. Dialog box types are defined in the header file `builder_.am`, located in the `install_dir/axdata/elf` directory.

-> [send_poke@](#)

Sends a poke message

Class DialogBoxClass set

Format this.send_poke@(code[, message])

Arguments code A unique number assigned to the poke message. code must be less than 10,000.

message The message to send. This value can be any ELF data type including a multi-level array.

Description Sends the message you specify. The message will be received by any dialog boxes set to accept the poke message. See [accept_pokes@](#) for information about setting the accepted poke codes.

-> title@

Sets a dialog box title

Class DialogBoxClass set

Format this.title@(argument)

Arguments argument A string for the dialog box title.

Description The set method title@ sets a dialog box title to a passed string. You can use the method with the get method title@ to verify and set the dialog box title.

For example, to verify a dialog box title use the following:

```
var title
value = this.title@           'get method
if title <> "Original Dialog Box Title"
this.title@("Original Dialog Box Title") 'set method
```

-> toggle_view_expressline@

Toggles dialog box ExpressLine display

Class DialogBoxClass set

Format this.toggle_view_expressline@

Description The set method toggle_view_expressline@ toggles a dialog box *ExpressLine* display.

-> type@

Sets dialog box type

Class DialogBoxClass set

Format this.type@(type)

Arguments type The dialog box type. The dialog box types are defined in the header file `builder_.am`, located in the `install_dir/axdata/elf` directory. The valid dialog box types are:

`DBOX_TYPE#NORMAL_` Normal. The standard dialog box type, only 1 instance of the dialog box is allowed at one time

`DBOX_TYPE#MAIN_` Main. The dialog box that appears when you run a Builder application. You can designate only one Main dialog box.

`DBOX_TYPE#RESPONSE_` Response. The application action is suspended, waiting for information from the dialog box. Application action resumes when the dialog box is dismissed.

`DBOX_TYPE#MULTIPLE_` Multiple. The dialog box can have multiple instances at the same time.

Description The set method `type@` sets the dialog box type.

For example, to get the dialog box type use the following:

```
var type
type = this.type@
```

-> `update@`

Updates widgets in an open dialog box

Class DialogBoxClass set

Format this.update@

Description The set method `update@` updates widgets in an open dialog box, redrawing all the information within the dialog box. The `update_event` is called for all dialog box widgets.

For example, to update a dialog box use the following:

```
this.update@
```

-> `update_children@`

Updates widgets in a dialog box

Class DialogBoxClass set

Format this.update_children@

Description The set method update_children@ updates widgets in a dialog box, redrawing all the information within the dialog box.

For example, to update a dialog box use the following:

```
this.update_children@
```

-> update_expressline@

Updates ExpressLine display in a dialog box

Class DialogBoxClass set

Format this.update_expressline@

Description The set method update_expressline@ updates the *ExpressLine* display in a dialog box, refreshing the icons in the *ExpressLine*.

-> variable_size_info@

Sets dialog box variable size information

Class DialogBoxClass set

Format this.variable_size_info@(minWidth, minHeight, maxWidth, maxHeight)

Arguments

minWidth	Minimum allowable width, in pixels.
minHeight	Minimum allowable height, in pixels.
maxWidth	Maximum allowable width, in pixels.
maxHeight	Maximum allowable height, in pixels.

Description The set method variable_size_info@ sets dialog box variable size information. If you want a resizable dialog box use the method to set the minimum and maximum resizable dimensions. Resizing a dialog box calls a *resize_event*.

This method must be set in the dialog box *initialize_event*, otherwise it will have no effect on the dialog box.

-> width@

Sets dialog box width

Class DialogBoxClass set

Format this.width@(newWidth)

Arguments newWidth The new dialog box width, in pixels.

Description The set method width@ sets the new dialog box width to the passed value. You can use this method with the get method width@ to verify and set a dialog box width.

For example, to set a dialog box width use the following:

```
var pixels
pixels = this.width@      'get method
IF pixels < 200
    this.width@(200)     'set method
```

-> x_pos@

Sets dialog box x-axis position

Class DialogBoxClass set

Format this.x_pos@(position)

Arguments position The x-axis position, in pixels, of the top left corner of the dialog box. If values for both x_pos@ and y_pos@ are 0, the dialog box is centered under the current mouse pointer location

Description The set method x_pos@ sets the new dialog box x-axis position on the screen. Use this method before the dialog box is displayed with an open@ method. You can use this method with the set method y_pos@ to set a dialog box position.

Since values of 0 for both x_pos@ and y_pos@ indicate that the dialog box be displayed under the current mouse pointer position, the closest that you can position a dialog box to the top left of a screen is position (1,0) or position (0,1).

If you use x_pos@ to set the horizontal position of the dialog box without using y_pos@ to set the vertical position, then the vertical position is set to 0. The maximum values for x_pos@ and y_pos@ will depend on the dimensions of your screen

For example, to set a dialog box to display at position (10,50) use the following:

```
this.x_pos@(10)
this.y_pos@(50)
this.open@
```

-> y_pos@

Sets dialog box y-axis position

Class DialogBoxClass set

Format this.y_pos@(position)

Arguments position The y-axis position, in pixels, of the top left corner of the dialog box. If values for both x_pos@ and y_pos@ are 0, the dialog box is centered under the current mouse pointer location

Description The set method y_pos@ sets the new dialog box y-axis position on the screen. Use this method before the dialog box is displayed with an open@ method. You can use this method with the set method x_pos@ to set a dialog box position.

Since values of 0 for both x_pos and y_pos indicate that the dialog box be displayed under the current mouse pointer position, the closest that you can position a dialog box to the top left of a screen is position (1,0) or position (0,1).

If you use y_pos@ to set the vertical position of the dialog box without using x_pos@ to set the horizontal position, then the horizontal position is set to 0. The maximum values for x_pos@ and y_pos@ will depend on the dimensions of your screen

For example, to set a dialog box to display at position (10,50) use the following:

```
this.x_pos@(10)
this.y_pos@(50)
this.open@
```

-> destroy_event

Called before destroying dialog box

Class DialogBoxClass event

Format this.destroy_event

Description The `destroy_event` is called by Applixware Builder before destroying a dialog box. Use the method to save or verify dialog box settings before closing and destroying the dialog box. This is a user-defined event, place all actions you want performed in the event definition.

<- `error_event`

Called for system errors

Class DialogBoxClass event

Format `flag = this.error_event(error_object)`

Arguments `error_object` An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

-> `initialize_event`

Called before posting application dialog box

Class DialogBoxClass event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

-> `poke_event`

Called when a poke message is received

Class DialogBoxClass event

Format this.poke_event(code, info)

Arguments code The poke code.
info The information passed with the poke code.

Description The poke_event is called by Applixware Builder when a dialog box receives a poke message. This is a user-defined event, place all actions you want performed in the event definition. Use the event to perform an action relating to the poke code and data passed in the message.

-> resize_event

Called when a dialog box is resized

Class DialogBoxClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments width The changed dialog box width.
height The changed dialog box height.
old_width The original dialog box width.
old_height The original dialog box height.

Description The resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

-> terminate_event

Called before closing or destroying dialog box

Class DialogBoxClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

-> time_out_event

Called on object timer time-out

Class DialogBoxClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the **timer@** method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

<- control_color@

Returns color used by object

Class EditBoxClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the edit box. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [EditBoxClass Methods](#) for more information.

<- cursor_line_number@

Returns line number of current cursor position

Class EditBoxClass get

Format line = this.cursor_line_number@

Description The get method cursor_line_number@ returns the the line number of the current cursor position in the edit box. Line numbers are 0-based in the edit box.

See [EditBoxClass Methods](#) for more information.

<- height@

Returns edit box height

Class EditBoxClass get

Format pixels = this.height@

Description The get method height@ returns the edit box height in pixels. Use this method with the set method height@ to verify and change the edit box's height.

For example, to make the edit box height at least 50 pixels use the following:

```
var pixels
pixels = this.height@           'get method
IF pixels < 50
    this.height@ = 50         'set method
```

See [EditBoxClass Methods](#) for more information.

<- is_mono_space@

Returns text font spacing

Class EditBoxClass get

Format flag = this.is_mono_space@

Description The get method `is_mono_space@` returns the text font spacing as a Boolean value. The method returns TRUE if the text font is a monospaced font. The method returns FALSE if the text font is a variable-spaced font.

See [EditBoxClass Methods](#) for more information.

<- is_read_only@

Returns read-only status

Class EditBoxClass get

Format flag = this.is_read_only@

Description The get method `is_read_only@` returns the read-only status as a Boolean value. The method returns TRUE if the information in the edit box is read-only. The method returns FALSE if the information in the edit box is editable.

See [EditBoxClass Methods](#) for more information.

<- selected_word@

Returns selected string

Class EditBoxClass get

Format word = this.selected_word@

Description The get method selected_word@ returns the selected string in the edit box. The method only returns a string on one line. If the selection spans multiple lines you must use the get method selection@. If you use this method with a multiple line selection an error is thrown.

For example, to get the selected word in the edit box you would use the method as follows:

```
var word
word = this.selected_word@
```

See [EditBoxClass Methods](#) for more information.

<- selection@

Returns start and end of selection

Class EditBoxClass get

Format array = this.selection@

Description The get method selection@ returns the selection in the edit box as a 4 element array. The array contains the following information:

array[0]	The beginning line number, 0-based.
array[1]	The beginning character position, 0-based.
array[2]	The end line number, 0-based.
array[3]	The end character position, 0-based.

See [EditBoxClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class EditBoxClass get

Format colors = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears in the edit box.

The color settings are returned in the following format:

colors[0]	The color type. The valid color types are:
1	RGB
2	Named color
3	Widget color
4	Work area color
5	HSB
6	CMYK

The following values apply to RGB color type:

colors[1]	The RGB red value.
colors[2]	The RGB blue value.
colors[3]	The RGB green value.

The following values apply to CMYK color type:

colors[1]	The CMYK cyan value.
colors[2]	The CMYK magenta value.
colors[3]	The CMYK yellow value.
colors[4]	The CMYK black value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [EditBoxClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class EditBoxClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the edit box. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
-----------	----------------

font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [EditBoxClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class EditBoxClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. Use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the edit box.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [EditBoxClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class EditBoxClass get

Format flag = this.text_font_italic@

Description The get method `text_font_italic@` returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. Use this method with the set method `text_font_italic@` to verify and change the object text italic state.

Text appears in the edit box.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [EditBoxClass Methods](#) for more information.

`<- text_font_name@`

Returns text font name

Class EditBoxClass get

Format name = this.text_font_name@

Description The get method `text_font_name@` returns the object text font name. Use this method with the set method `text_font_name@` to verify and change the object text font.

Text appears in the edit box.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"  'set method
```

See [EditBoxClass Methods](#) for more information.

`<- text_font_size@`

Returns text font size

Class EditBoxClass get

Format size = this.text_font_size@

Description The get method text_font_size@ returns the object text font size. Use this method with the set method text_font_size@ to verify and change the object text size.

Text appears in the edit box.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [EditBoxClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class EditBoxClass get

Format flag = this.text_is_shadowed@

Description The get method text_is_shadowed@ returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the edit box. Text only appears shadowed on color displays.

See [EditBoxClass Methods](#) for more information.

<- thickness@

Returns edit box thickness

Class EditBoxClass get

Format pixels = this.thickness@

Description The get method thickness@ returns the edit box thickness in pixels. The thickness is the bevel appearance of the edit box sides.

See [EditBoxClass Methods](#) for more information.

<- value@

Returns edit box contents

Class EditTextClass get

Format array = this.value@

Description The get method value@ returns the contents the edit box as an array of strings.

For example, to get the contents of the edit box use the following:

```
var array
array = this.value@
```

See [EditBoxClass Methods](#) for more information.

<- width@

Returns edit box width

Class EditTextClass get

Format pixels = this.width@

Description The get method width@ returns the edit box width in pixels. Use this method with the set method width @to verify and change the edit box's width.

For example, to make the edit box width at least 250 pixels use the following:

```
var pixels
pixels = this.width@           'get method
IF pixels< 250
    this.width@ = 250         'set method
```

See [EditBoxClass Methods](#) for more information.

<- word_at_cursor@

Returns string at cursor

Class EditBoxClass get

Format string = this.word_at_cursor@

Description The get method `word_at_cursor@` returns a string at the current cursor position in the edit box. The method returns the string to the right of the cursor, ignoring spaces. If the cursor is in a string, the string is returned.

For example, to get the string at the current cursor position in the edit box use the following:

```
var word
word = this.word_at_cursor@
```

See [EditBoxClass Methods](#) for more information.

-> button3_menu_info@

Sets pop up menu info

Class EditBoxClass set

Format this.button3_menu_info@(format arrayof rminfo@ info)

Arguments info An array of `rminfo@` information. The `rminfo@` format is defined in the `dialog_.am` file, located in the `install_dir/axdata/elf` directory. The `rminfo@` format is defined as:

format `rminfo@`

name, The name displayed in the menu.

macro_name, The macro or method name. Macro names must begin with the `@` character to indicate it is a macro, not a method.

args, An argument string.

active A Boolean value, TRUE means the menu option is enabled, FALSE means the menu option is grayed and disabled.

Description The set method `button3_menu_info@` sets the pop up menu information. A pop up menu appears with a right mouse button press. A pop up menu is a free-floating menu associated with a dialog box control. A pop up menu can be activated when the mouse pointer is in the control area.

See [EditBoxClass Methods](#) for more information.

-> `control_color@`

Sets control color

Class `EditBoxClass` set

Format `this.control_color@(type, c1, c2, c3, c4)`

Arguments `type` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- `c1` The RGB red value.
- `c2` The RGB blue value.
- `c3` The RGB green value.

The following values apply to CMYK color type:

- `c1` The CMYK cyan value.
- `c2` The CMYK magenta value.
- `c3` The CMYK yellow value.
- `c4` The CMYK black value.

The following values apply to HSB color type:

- `c1` The HSB hue value.

- c2 The HSB HSB value.
- c3 The CMYK brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values. See [EditBoxClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class EditBoxClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [EditBoxClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class EditBoxClass set

Format `this.control_color_is_workspace@(flag)`

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method `control_color_is_workspace@` sets the color used by the object. See [EditBoxClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class EditTextClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name.

See [EditTextClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class EditTextClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the control color with RGB values.

See [EditTextClass Methods](#) for more information.

-> copy_to_clipboard@

Copies selected text to clipboard

Class EditTextClass set

Format this.copy_to_clipboard@

Description The set method copy_to_clipboard@ copies the selected text in the edit box to the clipboard. Use the set method paste_from_clipboard@ to paste the copied text into the edit box.

See [EditBoxClass Methods](#) for more information.

-> **cut_to_clipboard@**

Cuts selected text from edit box to clipboard

Class EditBoxClass set

Format this.cut_to_clipboard@

Description The set method cut_to_clipboard@ cuts the selected text from the edit box and places it in the clipboard. Use the set method paste_from_clipboard@ to paste the copied text into the edit box.

See [EditBoxClass Methods](#) for more information.

-> **display@**

Displays edit box

Class EditBoxClass set

Format this.display@

Description The set method display@ displays the edit box in the dialog box.

See [EditBoxClass Methods](#) for more information.

-> **height@**

Sets edit box height

Class EditBoxClass set

Format this.height@(value)

Arguments value The height, in pixels, of the edit box.

Description The set method height@ sets the edit box's height. Use this method with the get method height@ to verify and change the edit box's height.

For example, to make the edit box height at least 50 pixels use the following:

```
var pixels
pixels = this.height@           'get method
IF pixels < 50
    this.height@ = 50         'set method
```

See [EditBoxClass Methods](#) for more information.

-> **is_mono_space@**

Sets text font spacing

Class EditBoxClass set

Format this.is_mono_space@(flag)

Arguments flag A Boolean value. TRUE makes the text font monospaced. FALSE makes the text font variable faced.

Description The set method is_mono_space@ sets the text font spacing.

See [EditBoxClass Methods](#) for more information.

-> **is_read_only@**

Sets read-only status

Class EditBoxClass set

Format this.is_read_only@(flag)

Arguments flag A Boolean value. TRUE makes the edit box read-only. FALSE makes the edit box editable.

Description The set method is_read_only@ sets the read-only status.

See [EditBoxClass Methods](#) for more information.

-> paste_from_clipboard@

Pastes text from clipboard into edit box

Class EditTextClass set

Format this.paste_from_clipboard@

Description The set method paste_from_clipboard@ pastes text from the clipboard into the edit box. Use the set methods [copy to clipboard@](#) and [cut to clipboard@](#) to place text into the clipboard.

See [EditTextClass Methods](#) for more information.

-> selection@

Sets selection in edit box

Class EditTextClass set

Format this.selection@(array)

Arguments array A 4-element array containing the following information:

array[0]	The beginning line number, 0-based.
array[1]	The beginning character position, 0-based.
array[2]	The end line number, 0-based.
array[3]	The end character position, 0-based.

Description The set method selection@ sets the selection in the edit box.

For example, to set the selection from line 0, character 0, to line 3, character 4 in the edit box you would use the method as follows:

```
var array
array = {0,0,3,4}
this.selection@(array)
```

See [EditTextClass Methods](#) for more information.

-> text_color@

Sets text color

Class EditBoxClass set

Format this.text_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method text_color@ sets the text color with an array of values. Text appears in the edit box.

See [EditBoxClass Methods](#) for more information.

-> text_color_cmyk@

Sets text CMYK color

Class EditTextClass set

Format this.text_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method text_color_cmyk@ sets the text color with CMYK values. Text appears in the edit box.

See [EditTextClass Methods](#) for more information.

-> text_color_name@

Sets text name color

Class EditTextClass set

Format this.text_color_name@(name)

Arguments name The color name.

Description The set method text_color_name@ sets the text color by name. Text appears in the edit box.

See [EditTextClass Methods](#) for more information.

-> text_color_rgb@

Sets text RGB color

Class EditTextClass set

Format this.text_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the edit box.

See [EditBoxClass Methods](#) for more information.

-> `text_cursor@`

Sets cursor position in edit box

Class EditBoxClass set

Format `this.text_cursor@(line, character)`

Arguments line The line number.
 character The character position number.

Description The set method `text_cursor@` sets the cursor position in the edit box to the passed line and character position.

For example, to set the cursor at line 1, character position 4 in the edit box you would use the method as follows:

```
    this.text_cursor@(1,4)
```

See [EditBoxClass Methods](#) for more information.

-> `text_font_attrs@`

Sets text font information

Class EditBoxClass set

Format `this.text_font_attrs@(format font_attrs_info@ fonts)`

Arguments fonts The font information:
 font_name The font name.
 font_size The font point size.

bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `text_font_attrs@` sets all the text font information. Text appears in the edit box. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [EditBoxClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class EditBoxClass set

Format `this.text_font_bold@(flag)`

Arguments flag Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.

Description The set method `text_font_bold@` sets the text bold state. Text appears in the edit box. Use this method with the get method `text_font_bold@` to verify and change the object text bold state. Use the set method `text_font_attrs@` to set all font information in one step.

See [EditBoxClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class EditBoxClass set

Format `this.text_font_italic@(flag)`

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method `text_font_italic@` sets the italic state of the object text. Text appears in the edit box. Use this method with the get method `text_font_italic@` to verify and

change the object text italic state. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [EditBoxClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class EditBoxClass set

Format `this.text_font_name@(name)`

Arguments name A string for the font.

Description The set method `text_font_name@` returns the object text font name. Text appears in the edit box. Use this method with the get method `text_font_name@` to verify and change the object text font. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [EditBoxClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class EditBoxClass set

Format `this.text_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `text_font_size@` sets the object text font size. Text appears in the edit box. Use this method with the get method `text_font_size@` to verify and change the object text size. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12        'set method
```

See [EditBoxClass Methods](#) for more information.

-> `text_is_shadowed@`

Sets text shadow state

Class EditBoxClass set

Format `this.text_is_shadowed@(flag)`

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method `text_is_shadowed@` sets the shadow state of the object text. Use this method with the get method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the edit box. Text only appears shadowed on color displays.

See [EditBoxClass Methods](#) for more information.

-> `thickness@`

Sets edit box thickness

Class EditBoxClass set

Format `this.thickness@(pixels)`

Arguments pixels The thickness, in pixels.

Description The set method `thickness@` sets the edit box thickness in pixels. The thickness is the bevel appearance of the edit box sides.

See [EditBoxClass Methods](#) for more information.

-> `value@`

Sets edit box contents

Class `EditBoxClass` `set`

Format `this.value@(valArray)`

Arguments `valArray` An array of strings.

Description The set method `value@` sets the edit box contents to a passed array of strings.

For example, to set the edit box contents you would use the method as follows:

```
this.value@ = {"First line", "Second Line", "Last Line"}
```

See [EditBoxClass Methods](#) for more information.

-> `width@`

Sets edit box width

Class `EditBoxClass` `set`

Format `this.width@(pixels)`

Arguments `pixels` The width, in pixels, of the edit box.

Description The set method `width@` sets the edit box's width. Use this method with the get method `width@` to verify and change the edit box's width.

For example, to make the edit box width at least 250 pixels use the following:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [EditBoxClass Methods](#) for more information.

<- button3_menu_state_event

Sets menu state before pop up menu is displayed

Class EditBoxClass event

Format flag = this.button3_menu_state_event(function)

Arguments function The name of a pop up menu function.

Description The button3_menu_state_event is called by Applixware Builder before a pop up menu is displayed for a control. The event should be programmed to return either a Boolean value or NULL. The event should return TRUE if the menu item for the function should be active. The event should return FALSE if the menu item for the function should be grayed and inactive. The event should return NULL if the state of the menu item is unchanged.

For example, the following event sets the state of different menu items:

```
get button3_menu_state_event(function)
    case of function
    case "menu_1", "menu_2"
        return(NULL) ' No change in status
    case "menu_3"
        return(TRUE) ' Active menu item
    case "menu_4"
        return(FALSE) ' Inactive menu item
    endcase
endget
```

Use the [button3_menu_info@](#) method to set the pop up menu information.

NOTE: The Debugger cannot access break points set in the button3_menu_state_event. Do not use break points in the button3_menu_state_event while you are debugging an application.

See [EditBoxClass Methods](#) for more information.

<- error_event

Called for system errors

Class EditTextClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [EditTextClass Methods](#) for more information.

-> changed_event

Called when edit box value changes

Class EditTextClass event

Format this.changed_event(value)

Arguments value An array of strings for the edit box value.

Description The changed_event is called by Applixware Builder when the edit box value changes. The event is passed the current value in the edit box as an array of strings.

The event is called when the focus changes from the edit box to another dialog box widget. This event is called before the [focus_out_event](#). This is a user-defined event, place all actions you want performed in the event definition.

See [EditTextClass Methods](#) for more information.

-> focus_in_event

Called when edit box receives focus

Class EditBoxClass event

Format this.focus_in_event

Description The focus_in_event is called by Applixware Builder when the edit box receives dialog box focus. This is a user-defined event, place all actions you want performed in the event definition.

See [EditBoxClass Methods](#) for more information.

-> focus_out_event

Called when edit box loses focus

Class EditBoxClass event

Format this.focus_out_event

Description The focus_out_event is called by Applixware Builder when the edit box loses dialog box focus. This is a user-defined event, place all actions you want performed in the event definition.

See [EditBoxClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class EditBoxClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the edit box dimensions, color, and so on.

See [EditBoxClass Methods](#) for more information.

-> **resize_event**

Called when a dialog box is resized

Class EditBoxClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The set event `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [EditBoxClass Methods](#) for more information.

-> **terminate_event**

Called before closing or destroying dialog box

Class EditBoxClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [EditBoxClass Methods](#) for more information.

-> **time_out_event**

Called on object timer time-out

Class EditBoxClass event

Format this.time_out_event

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the `timer@` method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [EditBoxClass Methods](#) for more information.

-> `typing_event`

Called when character typed in edit box

Class `EditBoxClass` event

Format `this.typing_event`

Description The `typing_event` is called by Applixware Builder when a character is typed in the edit box. This is a user-defined event, place all actions you want performed in the event definition.

See [EditBoxClass Methods](#) for more information.

-> `update_event`

Called to update dialog box control

Class `EditBoxClass` event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [EditBoxClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class EntryFieldClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the entry field. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [EntryFieldClass Methods](#) for more information.

<- cursor_pos@

Returns cursor position in entry field

Class EntryFieldClass get

Format pos = this.cursor_pos@

Description The get method cursor_pos@ returns the cursor position in an entry field. The left most position in an entry field is 1. If the method returns 0 the cursor is not in the entry field.

See [EntryFieldClass Methods](#) for more information.

<- is_numeric@

Returns numeric state

Class EntryFieldClass get

Format flag = this.is_numeric@

Description The get method is_numeric@ returns a Boolean value indicating the entry field numeric state. If the method returns TRUE the entry field only allows numeric input. If the method returns FALSE the entry field allows any character input.

See [EntryFieldClass Methods](#) for more information.

<- is_optional@

Returns optional state

Class EntryFieldClass get

Format flag = this.is_optional@

Description The get method is_optional@ returns a Boolean value indicating the entry field optional state. If the method returns TRUE an entry is optional in the entry field. If the method returns FALSE an entry must be placed in the entry field.

See [EntryFieldClass Methods](#) for more information.

<- is_password@

Returns password state

Class EntryFieldClass get

Format flag = this.is_password@

Description The get method is_password@ returns a Boolean value indicating the entry field password state. If the method returns TRUE the entry field is in password mode, characters typed in are not redisplayed. An asterisk is displayed for each character typed, The method returns FALSE if typed characters appear in the entry field.

See [EntryFieldClass Methods](#) for more information.

<- is_trimmed@

Returns trimmed state

Class EntryFieldClass get

Format flag = this.is_trimmed@

Description The get method is_trimmed@ returns a Boolean value indicating the entry field trimmed state. If the method returns TRUE leading and trailing spaces are removed from the returned value. If the method returns FALSE the value is returned unchanged.

See [EntryFieldClass Methods](#) for more information.

<- max_chars_length@

Returns the quantity of characters allowed

Class EntryFieldClass get

Format length = this.max_chars_length@

Description The get method max_chars_length@ returns the quantity of characters allowed in the entry field. You can use this method with the set method length to verify and change the entry field length.

For example, to set the entry field length to at least 20 characters you would use the method as follows:

```
IF this.max_chars_length@ < 20      'get method
    this.max_chars_length@ = 20      'set method
```

See [EntryFieldClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class EntryFieldClass get

Format colorArray = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears in the entry field.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.
colorArray[2] The HSB saturation value.
colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [EntryFieldClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class EntryFieldClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the entry field. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [EntryFieldClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class EntryFieldClass get

Format flag = this.text_font_bold@

Description The get method `text_font_bold@` returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. You can use this method with the set method `text_font_bold@` to verify and change the object text bold state.

Text appears in the entry field.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [EntryFieldClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class EntryFieldClass get

Format flag = this.text_font_italic@

Description The get method `text_font_italic@` returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method `text_font_italic@` to verify and change the object text italic state.

Text appears in the entry field.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@         'get method
IF NOT flag
    this.text_font_italic@ = TRUE     'set method
```

See [EntryFieldClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class EntryFieldClass get

Format name = this.text_font_name@

Description The get method text_font_name@ returns the object text font name. You can use this method with the set method text_font_name@ to verify and change the object text font. Text appears in the entry field.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"   'set method
```

See [EntryFieldClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class EntryFieldClass get

Format size = this.text_font_size@

Description The get method text_font_size@ returns the object text font size. You can use this method with the set method text_font_size@ to verify and change the object text size. Text appears in the entry field.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [EntryFieldClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class EntryFieldClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the entry field. Text only appears shadowed on color displays.

See [EntryFieldClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class EntryFieldClass get

Format colorArray = this.title_color@

Description The get method `title_color@` returns a two dimensional array of title color settings.

Title is the text that appears adjacent to the entry field.

`colorArray[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

`colorArray[1]` The RGB red value.

`colorArray[2]` The RGB blue value.

`colorArray[3]` The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [EntryFieldClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class EntryFieldClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name The font name.

font_size The font point size.

bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

shadow Not used.

See [EntryFieldClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class EntryFieldClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears adjacent to the entry field.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [EntryFieldClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class EntryFieldClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears adjacent to the entry field.

For example, to make the object title italic use the following:

```
var flag
flag = this.title_font_italic@         'get method
IF NOT flag
```

```
        this.title_font_italic@ = TRUE           'set method
```

See [EntryFieldClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class EntryFieldClass get

Format name = this.title_font_name@

Description The get method `title_font_name@` returns the object title font name. You can use this method with the set method `title_font_name@` to verify and change the object text font.

Title is the text that appears adjacent to the entry field.

For example, to set the object title font to Courier use the following:

```
var name
name = this.title_font_name@           'get method
IF name <> "Courier"
        this.title_font_name@ = "Courier"   'set method
```

See [EntryFieldClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class EntryFieldClass get

Format size = this.title_font_size@

Description The get method `title_font_size@` returns the object title font size. You can use this method with the set method `title_font_size@` to verify and change the object text size.

Title is the text that appears adjacent to the entry field.

For example, to set the object title size to 12 use the following:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
```

`this.title_font_size@ = 12`

'set method

See [EntryFieldClass Methods](#) for more information.

`<- title_is_shadowed@`

Returns title shadow state

Class EntryFieldClass get

Format `flag = this.title_is_shadowed@`

Description The get method `title_is_shadowed@` returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. Use this method with the set method `title_is_shadowed@` to verify and change the object text shadow state.

Title is the text that appears adjacent to the entry field. Text only appears shadowed on color displays.

See [EntryFieldClass Methods](#) for more information.

`<- title_position@`

Returns title position

Class EntryFieldClass get

Format `value = this.title_position@`

Description The get method `title_position@` returns the title position in relation to the entry field. The valid position values are:

-1	None
0	Left
1	Top

See [EntryFieldClass Methods](#) for more information.

<- valid_chars@

Returns acceptable characters

Class EntryFieldClass get

Format string = this.valid_chars@

Description The get method valid_chars@ returns the acceptable characters in the entry field as a string. For example, if the method returns the string "abc", the entry field only accepts a, b, or c as valid input characters.

See [EntryFieldClass Methods](#) for more information.

<- value@

Returns entry field value

Class EntryFieldClass get

Format value = this.value@

Description The get method value@ returns the entry field value as a string. You can use this method with the set method value@ to verify and change the entry field value.

For example, to set the entry field value to Entry Field Value you would use the method as follows:

```
IF this.value@ <> "Entry Field Value"           'get method
    this.value@ = "Entry Field Value" 'set method
```

See [EntryFieldClass Methods](#) for more information.

<- width@

Returns entry field width

Class EntryFieldClass get

Format value = this.width@

Description The get method `width@` returns the entry field width as the number of characters visible in the entry field. You can use this method with the set method `width@` to verify and change the entry field width.

For example, to set the entry field width to 15 you would use the method as follows:

<code>IF this.width@ < 15</code>	<code>'get method</code>
<code> this.width@ = 15</code>	<code>'set method</code>

See [EntryFieldClass Methods](#) for more information.

-> `button3_menu_info@`

Sets pop up menu info

Class EntryFieldClass set

Format `this.button3_menu_info@(format arrayof rminfo@ info)`

Arguments info An array of `rminfo@` information. The `rminfo@` format is defined in the `dialog_.am` file, located in the `install_dir/axdata/elf` directory. The `rminfo@` format is defined as:

<code>format rminfo@</code>	
<code>name,</code>	The name displayed in the menu.
<code>macro_name,</code>	The macro or method name. Macro names must begin with the <code>@</code> character to indicate it is a macro, not a method.
<code>args,</code>	An argument string.
<code>active</code>	A Boolean value, TRUE means the menu option is enabled, FALSE means the menu option is grayed and disabled.

Description The set method `button3_menu_info@` sets the pop up menu information. A pop up menu appears with a right mouse button press. A pop up menu is a free-floating menu associated with a dialog box control. A pop up menu can be activated when the mouse pointer is in the control area.

See [EntryFieldClass Methods](#) for more information.

-> control_color@

Sets control color

Class EntryFieldClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method control_color@ sets the control color with an array of values.

See [EntryFieldClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class EntryFieldClass set

Format this.control_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method control_color_cmyk@ sets the control color with CMYK values. See [EntryFieldClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class EntryFieldClass set

Format this.control_color_is_workspace@(flag)

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method control_color_is_workspace@ sets the color used by the object. See [EntryFieldClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class EntryFieldClass set

Format this.control_color_name@(name)

Arguments

name	The color name.
------	-----------------

Description The set method `control_color_name@` sets the control color by name.

See [EntryFieldClass Methods](#) for more information.

-> `control_color_rgb@`

Sets control RGB color

Class EntryFieldClass set

Format `this.control_color_rgb@(red, green, blue)`

Arguments

<code>red</code>	The RGB red value.
<code>green</code>	The RGB blue value.
<code>blue</code>	The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values.

See [EntryFieldClass Methods](#) for more information.

-> `cursor_pos@`

Sets cursor position in entry field

Class EntryFieldClass set

Format `this.cursor_pos@(pos)`

Arguments `pos` The cursor position from the left side of the entry field.

Description The set method `cursor_pos@` sets the cursor position in an entry field. The left most position in an entry field is 1. If the method sets 0 the cursor is not in the entry field.

See [EntryFieldClass Methods](#) for more information.

-> `display@`

Displays entry field

Class EntryFieldClass set

Format this.display@

Description The set method display@ displays the entry field in the dialog box.

See [EntryFieldClass Methods](#) for more information.

-> is_numeric@

Sets numeric state

Class EntryFieldClass set

Format this.is_numeric@(flag)

Arguments flag A Boolean value. TRUE allows only numeric input in the entry field.
FALSE allows any character input. The default is FALSE

Description The set method is_numeric@ sets the entry field numeric state.

See [EntryFieldClass Methods](#) for more information.

-> is_optional@

Sets optional state

Class EntryFieldClass set

Format this.is_optional@(flag)

Arguments flag A Boolean value. TRUE allows optional input in the entry field. FALSE re-
quires that an entry must be placed in the entry field. The default is
FALSE

Description The set method is_optional@ sets the entry field optional state.

See [EntryFieldClass Methods](#) for more information.

-> is_password@

Sets password state

Class EntryFieldClass set

Format this.is_password@(flag)

Arguments flag A Boolean value. TRUE places the entry field in password mode, characters typed in are not redisplayed. An asterisk is displayed for each character typed. FALSE allows typed characters to appear in the entry field. The default is FALSE.

Description The set method is_password@ sets the entry field password state.

See [EntryFieldClass Methods](#) for more information.

-> is_selected@

Sets entry field selection state

Class EntryFieldClass set

Format this.is_selected@(flag)

Arguments flag A Boolean value. TRUE selects the entire entry field contents FALSE does not select the entry field contents.

Description The set method is_selected@ sets the entry field selection state. Use the method to select the entry field contents before deleting them.

See [EntryFieldClass Methods](#) for more information.

-> is_trimmed@

Sets trimmed state

Class EntryFieldClass set

Format this.is_trimmed@(flag)

Arguments flag A Boolean value. TRUE removes leading and trailing spaces from the returned value, FALSE leaves the returned value unchanged.

Description The set method is_trimmed@ sets the entry field trimmed state.

See [EntryFieldClass Methods](#) for more information.

-> max_chars_length@

Sets maximum number of characters allowed in entry field

Class EntryFieldClass set

Format this.max_chars_length@(value)

Arguments value The number of characters.

Description The set method max_chars_length@ sets the maximum number of characters allowed in an entry field. You can use this method with the get method max_chars_length@ to verify and change the entry field length.

For example, to set the entry field length to at least 20 characters you would use the method as follows:

IF this.max_chars_length@ < 20	'get method
this.max_chars_length@ = 20	'set method

See [EntryFieldClass Methods](#) for more information.

-> text_color@

Sets text color

Class EntryFieldClass set

Format this.text_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- | | |
|----|---------------------|
| c1 | The RGB red value. |
| c2 | The RGB blue value. |

c3 The RGB green value.

The following values apply to CMYK color type:

c1 The CMYK cyan value.

c2 The CMYK magenta value.

c3 The CMYK yellow value.

c4 The CMYK black value.

The following values apply to HSB color type:

c1 The HSB hue value.

c2 The HSB saturation value.

c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears in the entry field.

See [EntryFieldClass Methods](#) for more information.

-> `text_color_cmyk@`

Sets text CMYK color

Class EntryFieldClass set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears in the entry field.

See [EntryFieldClass Methods](#) for more information.

-> text_color_name@

Sets text name color

Class EntryFieldClass set

Format this.text_color_name@(name)

Arguments name The color name.

Description The set method text_color_name@ sets the text color by name. Text appears in the entry field.

See [EntryFieldClass Methods](#) for more information.

-> text_color_rgb@

Sets text RGB color

Class EntryFieldClass set

Format this.text_color_rgb@(red, green, blue)

Arguments red The RGB red value.
green The RGB blue value.
blue The RGB green value.

Description The set method text_color_rgb@ sets the text color with RGB values. Text appears in the entry field.

See [EntryFieldClass Methods](#) for more information.

-> text_font_attrs@

Sets text font information

Class EntryFieldClass set

Format this.text_font_attrs@(format font_attrs_info@ fonts)

Arguments fonts The font information:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `text_font_attrs@` sets all the text font information. Text appears in the entry field. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [EntryFieldClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class EntryFieldClass set

Format `this.text_font_bold@(flag)`

Arguments flag Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.

Description The set method `text_font_bold@` sets the text bold state. Text appears in the entry field. You can use this method with the get method `text_font_bold@` to verify and change the object text bold state. Use the set method `text_font_attrs@` to set all font information in one step.

See [EntryFieldClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class EntryFieldClass set

Format `this.text_font_italic@(flag)`

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method `text_font_italic@` sets the italic state of the object text. Text appears in the entry field. You can use this method with the get method `text_font_italic@` to verify and change the object text italic state. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [EntryFieldClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class EntryFieldClass set

Format `this.text_font_name@(name)`

Arguments name A string for the font.

Description The set method `text_font_name@` returns the object text font name. Text appears in the entry field. You can use this method with the get method `text_font_name@` to verify and change the object text font. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [EntryFieldClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class EntryFieldClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears in the entry field. You can use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [EntryFieldClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class EntryFieldClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method text_is_shadowed@ sets the shadow state of the object text. Use this method with the get method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the entry field. Text only appears shadowed on color displays.

See [EntryFieldClass Methods](#) for more information.

-> thickness@

Sets entry field thickness

Class EntryFieldClass set

Format this.thickness@(pixels)

Arguments pixels The thickness, in pixels.

Description The set method thickness@ sets the entry field thickness in pixels. The thickness is the bevel appearance of the entry field sides.

See [EntryFieldClass Methods](#) for more information.

-> title_color@

Sets title color

Class EntryFieldClass set

Format this.title_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.

- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values. See [EntryFieldClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class EntryFieldClass set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values. See [EntryFieldClass Methods](#) for more information.

-> `title_color_name@`

Sets title name color

Class EntryFieldClass set

Format `this.title_color_name@(name)`

Arguments name The color name.

Description The set method `title_color_name@` sets the title color by name.

See [EntryFieldClass Methods](#) for more information.

-> title_color_rgb@

Sets title RGB color

Class EntryFieldClass set

Format this.title_color_rgb@(red, green, blue)

Arguments

red	The RGB red value.
green	The RGB blue value.
blue	The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [EntryFieldClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class EntryFieldClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method title_font_attrs@ sets all the title font information. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format.

See [EntryFieldClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class EntryFieldClass set

Format this.title_font_bold@(flag)

Arguments flag Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.

Description The set method title_font_bold@ sets the title bold state. You can use this method with the get method title_font_bold@ to verify and change the object title bold state. Use the set method title_font_attrs@ to set all font information in one step.

See [EntryFieldClass Methods](#) for more information.

-> title_font_italic@

Sets title italic state

Class EntryFieldClass set

Format this.title_font_italic@(flag)

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method title_font_italic@ sets the italic state of the object title. You can use this method with the get method title_font_italic@ to verify and change the object title italic state. Use the set method title_font_attrs@ to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [EntryFieldClass Methods](#) for more information.

-> title_font_name@

Sets title font name

Class EntryFieldClass set

Format this.title_font_name@(name)

Arguments name A string for the font.

Description The set method title_font_name@ returns the object title font name. You can use this method with the get method title_font_name@ to verify and change the object title font. Use the set method title_font_attrs@ to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [EntryFieldClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class EntryFieldClass set

Format this.title_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method title_font_size@ sets the object title font size. You can use this method with the get method title_font_size@ to verify and change the object title size. Use the set method title_font_attrs@ to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@         'get method
IF size <> 12
    this.title_font_size@ = 12       'set method
```

See [EntryFieldClass Methods](#) for more information.

-> title_is_shadowed@

Sets title shadow state

Class EntryFieldClass set

Format this.title_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method title_is_shadowed@ sets the shadow state of the object title. Use this method with the get method title_is_shadowed@ to verify and change the object title shadow state.

Title is the text that appears adjacent to the entry field. Text only appears shadowed on color displays.

See [EntryFieldClass Methods](#) for more information.

-> title_position@

Sets title position

Class EntryFieldClass set

Format this.title_position@(value)

Arguments value The title position value. The valid position values are:

-1	None
0	Left
1	Top

Description The set method title_position@ sets the title position in relation to the entry field.

See [EntryFieldClass Methods](#) for more information.

Format this.width@(value)

Arguments value The number of characters visible in the entry area.

Description The set method width@ sets the entry field width, the number of characters visible in the entry area. You can use this method with the set method width@ to verify and change the entry field width.

For example, to set the entry field width to 15 you would use the method as follows:

```
IF this.width@ < 15                    'get method
   this.width@ = 15                    'set method
```

See [EntryFieldClass Methods](#) for more information.

<- button3_menu_state_event

Sets menu state before pop up menu is displayed

Class EntryFieldClass event

Format flag = this.button3_menu_state_event(function)

Arguments function The name of a pop up menu function.

Description The button3_menu_state_event is called by Applixware Builder before a pop up menu is displayed for a control. The event should be programmed to return either a Boolean value or NULL. The event should return TRUE if the menu item for the function should be active. The event should return FALSE if the menu item for the function should be grayed and inactive. The event should return NULL if the state of the menu item is unchanged.

For example, the following event sets the state of different menu items:

```
get button3_menu_state_event(function)
  case of function
  case "menu_1", "menu_2"
    return(NULL) ' No change in status
  case "menu_3"
    return(TRUE) ' Active menu item
  case "menu_4"
    return(FALSE) ' Inactive menu item
```

```
        endcase
    endget
```

Use the [button3_menu_info@](#) method to set the pop up menu information.

NOTE: The Debugger cannot access break points set in the `button3_menu_state_event`. Do not use break points in the `button3_menu_state_event` while you are debugging an application.

See [EntryFieldClass Methods](#) for more information.

<- error_event

Called for system errors

Class EntryFieldClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [EntryFieldClass Methods](#) for more information.

-> changed_event

Called when entry field value changes

Class EntryFieldClass event

Format this.changed_event(value)

Arguments value A string for the entry field value.

Description The `changed_event` is called by Applixware Builder when the entry field value changes. The event is passed the the current value in the entry area as a string. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

-> `focus_in_event`

Called when entry field receives focus

Class `EntryFieldClass` event

Format `this.focus_in_event`

Description The `focus_in_event` is called by Applixware Builder when the entry field receives dialog box focus. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

-> `focus_out_event`

Called when entry field loses focus

Class `EntryFieldClass` event

Format `this.focus_out_event`

Description The `focus_out_event` is called by Applixware Builder when the entry field loses dialog box focus. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

-> `initialize_event`

Called before displaying a control

Class `EntryFieldClass` event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the entry field dimensions, color, and so on.

See [EntryFieldClass Methods](#) for more information.

-> `resize_event`

Called when a dialog box is resized

Class `EntryFieldClass` event

Format `this.resize_event(width, height, old_width, old_height)`

Arguments

<code>width</code>	The changed dialog box width.
<code>height</code>	The changed dialog box height.
<code>old_width</code>	The original dialog box width.
<code>old_height</code>	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [EntryFieldClass Methods](#) for more information.

-> `terminate_event`

Called before closing or destroying dialog box

Class `EntryFieldClass` event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class EntryFieldClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [EntryFieldClass Methods](#) for more information.

-> typing_event

Called when character typed in entry field

Class EntryFieldClass event

Format this.typing_event

Description The typing_event is called by Applixware Builder when a character is typed in the entry field. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class EntryFieldClass event

Format this.update_event

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [EntryFieldClass Methods](#) for more information.

`<- error_file@`

Returns the source file of the most recently thrown error

Class ErrorClass get

Format file = this.error_file@

Description The get method `error_file@` returns the source file of the most recently thrown error. Use the method after you catch an error in the [error_event](#).

See [ErrorClass Methods](#) for more information.

`<- error_function@`

Returns the last error method or macro name

Class ErrorClass get

Format function = this.error_function@

Description The get method `error_function@` returns the name of the macro or method that caused the last error throw to occur. Use the method after you catch an error in the [error_event](#).

See [ErrorClass Methods](#) for more information.

`<- error_line@`

Returns the line number of the most recently thrown error

Class ErrorClass get

Format line = this.error_line@

Description The get method error_line@ returns the line number of the most recently known error. Use the get method [error_file@](#) to retrieve the file containing the error. Use the method after you catch an error in the [error event](#).

See [ErrorClass Methods](#) for more information.

<- error_number@

Returns the error code of the most recently thrown error

Class ErrorClass get

Format number = this.error_number@

Description The get method error_number@ returns the error code of the most recently thrown error. Use the method after you catch an error in the [error event](#).

See [ErrorClass Methods](#) for more information.

<- error_object@

Returns object handle in which error occurred

Class ErrorClass get

Format object obj = this.error_object@

Description The get method error_object@ returns the object handle of the object in which the error occurred. The object is passed to the application's [error event](#), which you program to handle the error. The error object contains the following information:

error_file

error_function

error_line

error_number

error_prepend

error_string

Use the method after you catch an error.

See [ErrorClass Methods](#) for more information.

<- error_prepend@

Returns the item to which the error applies

Class ErrorClass get

Format string = this.error_prepend@

Description The get method error_prepend@ returns a string that represents the item to which the error applies. For example, if an error is generated because a file cannot be opened for writing, the string returned by error_prepend@ could be the name of the file that cannot be opened for writing. Use the method after you catch an error in the [error event](#).

See [ErrorClass Methods](#) for more information.

<- error_string@

Returns the string value of the most recently thrown error

Class ErrorClass get

Format number = this.error_string@

Description The get method error_string@ returns the string value of the most recently thrown error. The string value is associated with the error number returned by the get method [error_number@](#). Use the method after you catch an error in the [error event](#).

See [ErrorClass Methods](#) for more information.

<- error_event

Called for system errors

Class ErrorClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ErrorClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class `ErrorClass event`

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ErrorClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class `ErrorClass event`

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [ErrorClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class ErrorClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [ErrorClass Methods](#) for more information.

<- date_order@

Returns order in which data is returned

Class HistoricalDataClass get

Format val = this.date_order@

Description The get method date_order@ returns the order in which the historical data is returned. The method returns 0 if the date order is ascending (older to new) . The method returns 1 if the date order is descending (new to older).

See [HistoricalDataClass Methods](#) for more information.

<- end_date@

Returns end date for data search

Class HistoricalDataClass get

Format date = this.end_date@

Description The get method end_date@ returns the end date for the data search. Use the set methods start_date@ and end_date@ to establish the range of information to search.

See [HistoricalDataClass Methods](#) for more information.

<- fields@

Returns field names

Class HistoricalDataClass get

Format fieldNameArray = this.fields@

Description The get method field_names_list@ returns the names of the fields used to retrieve the historical data. The field names are returned as an array of strings.

See [HistoricalDataClass Methods](#) for more information.

<- period@

Returns time period for record entry search

Class HistoricalDataClass get

Format type = this.period@

Description The get method period@ returns the time period for the record entry search as a string. The valid time periods for a search are:

Daily

Weekly

Monthly

Quarterly

Yearly

See [HistoricalDataClass Methods](#) for more information.

<- query@

Queries for historical data and returns results

Class HistoricalDataClass get

Format results = this.query@(service, record, period, start_date, end_date, tossNullFlag, date_order, showFieldFlag)

Arguments

service	The name of the data feed service for example IDN_SELECTFEED.
record	The name of the record, for example APLX.O.
period	The string for the time period. The valid time period strings are: Daily Weekly Monthly Quarterly Yearly
start_date	The start date for the search, for example 6/1/95.
end_date	The end date for the search. If a date is not supplied and not set with set method end_date@ the current date is used.
tossNullFlag	A Boolean value, set to TRUE to exclude NULL points, FALSE to include NULL points.
date_order	The order in which the historical data is returned. Pass 0 for ascending date order(older to new), or 1 for descending date order (new to older)
showFieldFlag	A Boolean value, set to TRUE to show field names, FALSE to display only data.

Description The get method query@ initiates the historical data query and returns the results in a two-dimensional array. The passed arguments override any arguments set by the previous query or by the individual methods. If any of the arguments are not supplied or are set to NULL the values set by previous queries or by the individual methods are used.

See [HistoricalDataClass Methods](#) for more information.

<- record@

Returns the record name

Class HistoricalDataClass get

Format rec = this.record@

Description The get method record@ returns the record name used to search for historical data, for example APLX.O.

See [HistoricalDataClass Methods](#) for more information.

<- service@

Returns the service name

Class HistoricalDataClass get

Format service = this.service@

Description The get method record@ returns the data feed service name used to search for historical data, for example IDN_SELECTFEED.

See [HistoricalDataClass Methods](#) for more information.

<- show_field_names@

Returns the show field setting

Class HistoricalDataClass get

Format showFlag = this.show_field_names@

Description The get method show_field_names@ returns the show field setting as a Boolean value. The method returns TRUE if the field names are displayed with the data in the table, FALSE if only the data is displayed.

See [HistoricalDataClass Methods](#) for more information.

<- start_date@

Returns start date for data search

Class HistoricalDataClass get

Format date = this.start_date@

Description The get method start_date@ returns the start date for the data search. Use the set methods start_date@ and end_date@ to establish the range of information to search.

See [HistoricalDataClass Methods](#) for more information.

<- toss_null_points@

Returns the discard NULL information setting

Class HistoricalDataClass get

Format tossFlag = this.toss_null_points@

Description The get method toss_null_points@ returns the discard NULL information setting as a Boolean value. The method returns TRUE if NULL information is discarded, FALSE if NULL information is returned by the query.

See [HistoricalDataClass Methods](#) for more information.

<- widgets@

Returns controls receiving results

Class HistoricalDataClass get

Format object ctrl = this.widgets@

Description The get method widgets@ returns the controls receiving the results of the query. TableClass and ChartClass controls are the only controls that can receive the results of a historical data query.

See [HistoricalDataClass Methods](#) for more information.

-> add_field@

Sets field names

Class HistoricalDataClass set

Format this.add_field@(field)

Arguments field A string or array of field names.

Description The set method add_field@ sets field names for the historical data query.

See [HistoricalDataClass Methods](#) for more information.

-> add_widget@

Adds a control

Class HistoricalDataClass set

Format this.add_widget@(object ctrl)

Arguments ctrl A dialog box control or array of dialog box controls. TableClass and ChartClass controls are the only controls that can receive the results of a historical data query.

Description The set method add_widget@ associates the control with the historical data query.

See [HistoricalDataClass Methods](#) for more information.

-> date_order@

Sets order in which data is returned

Class HistoricalDataClass set

Format this.date_order@(val)

Arguments val The order value. The valid order values are:

- 0 Ascending (older to new)
- 1 Descending (new to older)

Description The set method `date_order@` sets the order in which the historical data is returned. See [HistoricalDataClass Methods](#) for more information.

-> `end_date@`

Sets end date for data search

Class HistoricalDataClass set

Format `this.end_date@(date)`

Arguments `date` The date as a string, for example 6/1/96. The date format must be a valid format as defined in the header file `install_dir/axdata/elf/datetim_.am`.

Description The set method `end_date@` sets the end date for the data search. Use the set methods `start_date@` and `end_date@` to establish the range of information to search.

See [HistoricalDataClass Methods](#) for more information.

-> `fields@`

Sets field names

Class HistoricalDataClass set

Format `this.fields@(fields)`

Arguments `field` A string or array of field names.

Description The set method `fields@` sets the names of the fields used to retrieve the historical data.

See [HistoricalDataClass Methods](#) for more information.

-> `period@`

Sets time period for record entry search

Class HistoricalDataClass set

Format `this.period@(type)`

Arguments type The timeperiod,as a string. The valid period strings are:
Daily
Weekly
Monthly
Quarterly
Yearly

Description The set method `period@` sets the time period for the record entry search to the passed string.

See [HistoricalDataClass Methods](#) for more information.

-> `query@`

Queries for historical data

Class HistoricalDataClass set

Format `this.query@(service, record, period, start_date, end_date, tossNullFlag, date_order, showFieldFlag)`

Arguments

service	The name of the data feed service for example IDN_SELECTFEED.
record	The name of the record, for example APLX.O.
period	The string for the time period. The valid time period strings are: Daily Weekly Monthly Quarterly Yearly
start_date	The start date for the search, for example 6/1/95.
end_date	The end date for the search. If a date is not supplied and not set with set method <code>end_date@</code> the current date is used.
tossNullFlag	A Boolean value, set to TRUE to exclude NULL points, FALSE to include NULL points.
date_order	The order in which the historical data is returned. Pass 0 for ascending date order(older to new), or 1 for descending date order (new to older)

showFieldFlag A Boolean value, set to TRUE to show field names, FALSE to display only data.

Description The set method query@ initiates the historical data query and updates all associated controls with the results. The passed arguments override any arguments set by the previous query or by the individual methods. If any of the arguments are not supplied or are set to NULL the values set by previous queries or by the individual methods are used.

See [HistoricalDataClass Methods](#) for more information.

-> record@

Sets the record name

Class HistoricalDataClass set

Format this.record@(rec)

Arguments rec The name of the record, for example APLX.O.

Description The set method record@ sets the record name used to search for historical data.

See [HistoricalDataClass Methods](#) for more information.

-> service@

Sets the service name

Class HistoricalDataClass set

Format this.service@(service)

Arguments service The name of the data feed service for example IDN_SELECTFEED.

Description The set method record@ sets the data feed service name used to search for historical data.

See [HistoricalDataClass Methods](#) for more information.

-> show_field_names@

Sets the show field setting

Class HistoricalDataClass set

Format this.show_field_names@(showFlag)

Arguments showFlag A Boolean value, set to TRUE to show field names, FALSE to display only data.

Description The set method show_field_names@ sets the show field setting.

See [HistoricalDataClass Methods](#) for more information.

-> start_date@

Sets start date for data search

Class HistoricalDataClass set

Format this.start_date@(date)

Arguments date The date as a string, for example 6/1/96. The date format must be a valid format as defined in the header file *install_dir/axdata/elf/datetim_.am*.

Description The set method start_date@ sets the start date for the data search. Use the set methods start_date@ and end_date@ to establish the range of information to search.

See [HistoricalDataClass Methods](#) for more information.

-> toss_null_points@

Returns the discard NULL information setting

Class HistoricalDataClass set

Format this.toss_null_points@(tossFlag)

Arguments tossFlag A Boolean value, set to TRUE to exclude NULL points, FALSE to include NULL points.

Description The set method `toss_null_points@` sets the discard NULL information setting. See [HistoricalDataClass Methods](#) for more information.

-> `widgets@`

Sets controls receiving results

Class HistoricalDataClass set

Format `this.widgets@(object ctrl)`

Arguments `ctrl` A dialog box control or array of dialog box controls. TableClass and ChartClass controls are the only controls that can receive the results of a historical data query.

Description The set method `widgets@` associates the control with the historical data query. See [HistoricalDataClass Methods](#) for more information.

<- `value@`

Returns bitmap file name

Class IconClass get

Format `name = this.value@`

Description The get method `value@` returns the bitmap file name as a string, without the bitamp .im file extension. The bitmap file is located in your ELF search path, or loaded as an application resource. You can use this method with the set method `value@` to verify and change the bitmap file.

For example, to use the bitmap file Bitmap.im use the following:

```
var name
name = this.value@           'get method
IF name< > "Bitmap"
    this.value@ = "Bitmap"   'set method
```

See [IconClass Methods](#) for more information.

-> display@

Displays the object

Class IconClass set

Format this.display@

Description The set method display@ displays the object in the dialog box. Use the method to update the display of an object after suspending the display with the set method display_suspend@. Use the methods to update the dialog box widgets at one time, instead of individually.

See [IconClass Methods](#) for more information.

-> value@

Sets bitmap file name

Class IconClass set

Format this.value@(name)

Arguments name A string for the bitmap file, without the bitmap .im file extension. The bitmap file must be located in your ELF search path, or loaded as an application resource.

Description The set method value@ sets the bitmap file name. You can use this method with the get method value@ to verify and change the bitmap file name.

For example, to use the bitmap file Bitmap.im you would use the method as follows:

```
var name
name = this.value@           'get method
IF name < > "Bitmap"
    this.value@ = "Bitmap"   'set method
```

See [IconClass Methods](#) for more information.

<- error_event

Called for system errors

Class IconClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [IconClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class IconClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the icon dimensions, position, and so on.

See [IconClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class IconClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [IconClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class IconClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [IconClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class IconClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [IconClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class IconClass event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [IconClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class LabelClass get

Format `colorArray = this.title_color@`

Description The get method `title_color@` returns a two dimensional array of title color settings.

Title is the text that appears in the label.

The color settings are returned in the following format:

`colorArray[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [LabelClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class LabelClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/daxdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name The font name.

font_size The font point size.

bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

shadow The font shadow state as a Boolean value, TRUE means the font is shadowed, otherwise it sets FALSE.

See [LabelClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class LabelClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears in the label.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [LabelClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class LabelClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears in the label.

For example, to make the object title italic use the following:

```

var flag
flag = this.title_font_italic@           'get method
IF NOT flag
    this.title_font_italic@ = TRUE       'set method

```

See [LabelClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class LabelClass get

Format name = this.title_font_name@

Description The get method `title_font_name@` returns the object title font name. You can use this method with the set method `title_font_name@` to verify and change the object text font.

Title is the text that appears in the label.

For example, to set the object title font to Courier use the following:

```

var name
name = this.title_font_name@           'get method
IF name <> "Courier"
    this.title_font_name@ = "Courier"  'set method

```

See [LabelClass Methods](#) for more information.

<- title_font_shadow@

Returns title shadow type

Class LabelClass get

Format type = this.title_font_shadow@

Description The get method `title_font_shadow@` returns the title shadow type. An object title is shadowed when the [title is shadowed@](#) method is set to TRUE. The valid shadow types are:

Title is the text that appears in the label.

See [LabelClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class LabelClass get

Format size = this.title_font_size@

Description The get method title_font_size@ returns the object title font size. You can use this method with the set method title_font_size@ to verify and change the object text size.

Title is the text that appears in the label.

For example, to set the object title size to 12 use the following:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [LabelClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class LabelClass get

Format flag = this.title_is_shadowed@

Description The get method title_is_shadowed@ returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. You can use this method with the set method title_is_shadowed@ to verify and change the object title shadow state.

Title is the text that appears in the label.

See [LabelClass Methods](#) for more information.

<- value@

Returns label text

Class LabelClass get

Format text = this.value@

Description The get method value@ returns the label text.

See [LabelClass Methods](#) for more information.

-> title_color@

Sets title color

Class LabelClass set

Format this.title_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values. See [LabelClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class LabelClass set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values. See [LabelClass Methods](#) for more information.

-> `title_color_name@`

Sets title name color

Class LabelClass set

Format `this.title_color_name@(name)`

Arguments name The color name.

Description The set method `title_color_name@` sets the title color by name. See [LabelClass Methods](#) for more information.

-> title_color_rgb@

Sets title RGB color

Class LabelClass set

Format this.title_color_rgb@(red, green, blue)

Arguments

red	The RGB red value.
green	The RGB blue value.
blue	The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [LabelClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class LabelClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	The font shadow state as a Boolean value, TRUE means the font is shadowed, otherwise it sets FALSE.

Description The set method title_font_attrs@ sets all the title font information. The font_attrs_info@ format is defined in the *install_dir/daxdata/elf/builder_.am* file. Include this file in any sources using the format.

See [LabelClass Methods](#) for more information.

-> title_font_name@

Sets title font name

Class LabelClass set

Format this.title_font_name@(name)

Arguments name A string for the font.

Description The set method title_font_name@ returns the object title font name. You can use this method with the get method title_font_name@ to verify and change the object title font. Use the set method title_font_attrs@ to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [LabelClass Methods](#) for more information.

-> title_font_shadow@

Sets title shadow type

Class LabelClass set

Format this.title_font_shadow@(type)

Arguments type The shadow type. The defined shadow types are:

Description The set method title_font_shadow@ sets the object title shadow type. An object title is shadowed when the [title is shadowed@](#) method is set to TRUE.

Title is the text that appears in the label.

See [LabelClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class LabelClass set

Format this.title_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method title_font_size@ sets the object title font size. You can use this method with the get method title_font_size@ to verify and change the object title size. Use the set method title_font_attrs@ to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [LabelClass Methods](#) for more information.

-> title_is_shadowed@

Sets title shadow state

Class LabelClass set

Format this.title_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method title_is_shadowed@ sets the shadow state of the object title. You can use this method with the get method title_is_shadowed@ to verify and change the object title shadow state.

Title is the text that appears in the label.

See [LabelClass Methods](#) for more information.

-> value@

Sets label text

Class LabelClass set

Format this.value@(text)

Arguments text A string that appears as the label text..

Description The set method value@ sets the label text.

See [LabelClass Methods](#) for more information.

<- error_event

Called for system errors

Class LabelClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [LabelClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class LabelClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the label size, color, and so on.

See [LabelClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class LabelClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [LabelClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class LabelClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [LabelClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class LabelClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [LabelClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class LabelClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [LabelClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class ListBoxClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the list box. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

<- force_list_item_display@

Returns the force_list_item_display flag

Class ListBoxClass get

Format flag = this.force_list_item_display@

Description Returns the current setting of the force_list_item_display flag. If this flag is set to TRUE, the list item specified by the List Box set method list_item_for_display@ is displayed at the top of the list box.

<- height@

Returns list box height

Class ListBoxClass get

Format lines = this.height@

Description The get method height@ returns the list box height in lines. Use this method with the set method height@ to verify and change the list box's height.

For example, to make the list box height at least 5 lines use the following:

```
var lines
lines = this.height@           'get method
IF lines < 5
    this.height@ = 5           'set method
```

<- hscroll_is_enabled@

Returns horizontal scroll bar status

Class ListBoxClass get

Format flag = this.hscroll_is_enabled@

Description The get method hscroll_is_enabled@ returns the horizontal scroll bar status as a Boolean value. The method returns TRUE if the horizontal scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

<- hscroll_length@

Returns horizontal scroll bar length

Class ListBoxClass get

Format pixels = this.hscroll_length@

Description The get method hscroll_length@ returns the horizontal scroll bar length, in pixels.

<- hscroll_origin@

Returns horizontal scroll bar origin

Class ListBoxClass get

Format pixels = this.hscroll_origin@

Description The get method hscroll_origin@ returns the horizontal scroll bar origin, in pixels, from the bottom left corner of the list box.

<- is_mono_space@

Returns text font spacing

Class ListBoxClass get

Format flag = this.is_mono_space@

Description The get method is_mono_space@ returns the text font spacing as a Boolean value. The method returns TRUE if the text font is a monospaced font. The method returns FALSE if the text font is a variable-spaced font.

<- is_multi_select@

Returns multi-select status

Class ListBoxClass get

Format flag = this.is_multi_select@

Description The get method `is_multi_select@` returns the multi-select status as a Boolean value. The method returns TRUE if the list box allows multiple selections. The method returns FALSE if the allows only single line selection.

`<- is_read_only@`

Returns read-only status

Class ListBoxClass get

Format `flag = this.is_read_only@`

Description The get method `is_read_only@` returns the read-only status as a Boolean value. The method returns TRUE if the information in the list box is read-only. The method returns FALSE if the information in the list box is editable.

`<- list_item_for_display@`

Returns the index of a list item to be displayed

Class ListBoxClass get

Format `itemNumber = this.list_item_to_display@`

Description Returns the index of the list item that is displayed when the `force_list_item_display@` flag is set to TRUE.

`<- text_color@`

Returns text color settings

Class ListBoxClass get

Format `colors = this.text_color@`

Description The get method `text_color@` returns a two dimensional array of text color settings. Text appears in the list box.

The color settings are returned in the following format:

`colors[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- colors[1] The RGB red value.
- colors[2] The RGB blue value.
- colors[3] The RGB green value.

The following values apply to CMYK color type:

- colors[1] The CMYK cyan value.
- colors[2] The CMYK magenta value.
- colors[3] The CMYK yellow value.
- colors[4] The CMYK black value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

<- text_font_attrs@

Returns text font information

Class ListBoxClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the list box. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

- font_name The font name.
- font_size The font point size.
- bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

<- text_font_bold@

Returns text bold state

Class ListBoxClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. Use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the list box.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

<- text_font_italic@

Returns text italic state

Class ListBoxClass get

Format flag = this.text_font_italic@

Description The get method text_font_italic@ returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. Use this method with the set method text_font_italic@ to verify and change the object text italic state.

Text appears in the list box.

For example, to make the object text italic use the following:

```
var flag
```

```
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

<- text_font_name@

Returns text font name

Class ListBoxClass get

Format name = this.text_font_name@

Description The get method text_font_name@ returns the object text font name. Use this method with the set method text_font_name@ to verify and change the object text font.

Text appears in the list box.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"  'set method
```

<- text_font_size@

Returns text font size

Class ListBoxClass get

Format size = this.text_font_size@

Description The get method text_font_size@ returns the object text font size. Use this method with the set method text_font_size@ to verify and change the object text size.

Text appears in the list box.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

<- text_is_shadowed@

Returns text shadow state

Class ListBoxClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the list box. Text only appears shadowed on color displays.

<- text_strings@

Returns list box contents

Class ListBoxClass get

Format valArray = this.text_strings@

Description The get method `text_strings@` returns the list box contents as an array of strings. Use this method with the set method `text_strings@` to verify and change the list box contents.

For example, to set the list box contents you would use the method as follows:

IF this.text_strings@ = NULL	'get method
this.text_strings@ = {"red", "white", "blue"}	'set method

<- thickness@

Returns list box thickness

Class ListBoxClass get

Format pixels = this.thickness@

Description The get method `thickness@` returns the list box thickness in pixels. The thickness is the bevel appearance of the list box sides.

<- value@

Returns current list box choice

Class ListBoxClass get

Format valueArray = this.value@

Description The get method value@ returns the current list box choice as a string or an array of strings. The method returns a string if is_multi_select@ is set to FALSE. The method returns an array of strings if is_multi_select@ is set to TRUE.

<- value_index@

Returns array index of list box choices

Class ListBoxClass get

Format indexArray = this.value_index@

Description The get method value_index@ returns the array index of the current list box choice as an array of numeric values. The method returns an integer, not an array, for a list box that is not multi-select. The list box text strings are a 0-based array.

If there are no selections in the list box the method returns -1, as a single-element array for a multi-select dialog box, or as an integer for a list box that is not multi-select. Use the method with the get method text_strings@ to reference the list box contents.

<- vscroll_is_enabled@

Returns vertical scroll bar status

Class ListBoxClass get

Format flag = this.vscroll_is_enabled@

Description The get method vscroll_is_enabled@ returns the vertical scroll bar status as a Boolean value. The method returns TRUE if the vertical scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

<- vscroll_length@

Returns vertical scroll bar length

Class ListBoxClass get

Format pixels = this.vscroll_length@

Description The get method vscroll_length@ returns the vertical scroll bar length, in pixels.

<- vscroll_origin@

Returns vertical scroll bar length

Class ListBoxClass get

Format pixels = this.vscroll_origin@

Description The get method vscroll_origin@ returns the vertical scroll bar origin, in pixels, from the top right corner of the list box.

<- width@

Returns list box width

Class ListBoxClass get

Format chars = this.width@

Description The get method width@ returns the list box width in characters. Use this method with the set method width@ to verify and change the list box's width.

For example, to make the list box width at least 25 characters use the followings:

```
var chars
```

```
chars = this.width@           'get method
```

```
IF chars < 25
```

```
    this.width@ = 25         'set method
```

-> button3_menu_info@

Sets pop up menu info

Class ListBoxClass set

Format this.button3_menu_info@(format arrayof rminfo@ info)

Arguments info An array of rminfo@ information. The rminfo@ format is defined in the dialog_.am file, located in the *install_dir/axdata/elf* directory. The rminfo@ format is defined as:

format rminfo@

name,	The name displayed in the menu.
macro_name,	The macro or method name. Macro names must begin with the @ character to indicate it is a macro, not a method.
args,	An argument string.
active	A Boolean value, TRUE means the menu option is enabled, FALSE means the menu option is grayed and disabled.

Description The set method button3_menu_info@ sets the pop up menu information. A pop up menu appears with a right mouse button press. A pop up menu is a free-floating menu associated with a dialog box control. A pop up menu can be activated when the mouse pointer is in the control area.

See [ListBoxClass Methods](#) for more information.

-> control_color@

Sets control color

Class ListBoxClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

1	RGB
2	Named color

- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values.

See [ListBoxClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class ListBoxClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [ListBoxClass Methods](#) for more information.

-> `control_color_is_workspace@`

Sets color used by object

Class ListBoxClass set

Format `this.control_color_is_workspace@(flag)`

Arguments flag Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.

Description The set method `control_color_is_workspace@` sets the color used by the object. See [ListBoxClass Methods](#) for more information.

-> `control_color_name@`

Sets control name color

Class ListBoxClass set

Format `this.control_color_name@(name)`

Arguments name The color name.

Description The set method `control_color_name@` sets the control color by name. See [ListBoxClass Methods](#) for more information.

-> `control_color_rgb@`

Sets control RGB color

Class ListBoxClass set

Format `this.control_color_rgb@(red, green, blue)`

Arguments red The RGB red value.

green The RGB blue value.
blue The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values.
See [ListBoxClass Methods](#) for more information.

-> `force_list_item_display@`

Forces a specified item to be displayed

Class `ListBoxClass` set

Format `this.force_list_item_display@(flag)`

Arguments `flag` A Boolean. If TRUE the list box will always display the list item specified by the `list_item_for_display@` method. If FALSE, the list box displays the first selected item at all times.

Description When FLAG is TRUE, the list box displays items as follows:

7. The list box displays the item specified by the `list_item_for_display@` method at the top of the list box.
8. If no item is specified for display, the list box displays the first selected item established by the `value_index@` method.

This flag should be established in the `initialize_event` for the list box.

-> `height@`

Sets list box height

Class `ListBoxClass` set

Format `this.height@(value)`

Arguments `value` The height, in lines, of the list box.

Description The set method `height@` sets the list box's height. Use this method with the get method `height@` to verify and change the list box's height.

For example, to make the list box height at least 5 lines use the following:

```
var lines
lines = this.height@           'get method
IF pixels < 5
    this.height@ = 5         'set method
```

See [ListBoxClass Methods](#) for more information.

-> **hscroll_is_enabled@**

Sets horizontal scroll bar status

Class ListBoxClass set

Format this.hscroll_is_enabled@(flag)

Arguments flag A Boolean value. TRUE makes the horizontal scroll bar enabled and visible. FALSE makes the scroll bar disabled and hidden.

Description The set method hscroll_is_enabled@ sets the horizontal scroll bar status.

See [ListBoxClass Methods](#) for more information.

-> **hscroll_length@**

Sets horizontal scroll bar length

Class ListBoxClass set

Format this.hscroll_length@(pixels)

Arguments pixels The length of the scroll bar, in pixels

Description The set method hscroll_length@ sets the horizontal scroll bar length.

See [ListBoxClass Methods](#) for more information.

-> **hscroll_origin@**

Sets horizontal scroll bar origin

Class ListBoxClass set

Format this.hscroll_origin@(pixels)

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the list box.

Description The set method hscroll_origin@ sets the horizontal scroll bar origin.

See [ListBoxClass Methods](#) for more information.

-> **is_mono_space@**

Sets text font spacing

Class ListBoxClass set

Format this.is_mono_space@(flag)

Arguments flag A Boolean value. TRUE makes the text font monospaced. FALSE makes the text font variable faced.

Description The set method is_mono_space@ sets the text font spacing.

See [ListBoxClass Methods](#) for more information.

-> **is_multi_select@**

Sets multi-select status

Class ListBoxClass set

Format this.is_multi_select@(flag)

Arguments flag A Boolean value. TRUE makes the list box multi-select. FALSE makes the list box single-select.

Description The set method `is_multi_select@` sets the multi-select status. A multi-select list box allows you to choose multiple list box items using the CTRL or SHIFT button and the left mouse button.

See [ListBoxClass Methods](#) for more information.

-> `is_read_only@`

Sets read-only status

Class ListBoxClass set

Format `this.is_read_only@(flag)`

Arguments flag A Boolean value. TRUE makes the list box read-only. FALSE makes the list box editable.

Description The set method `is_read_only@` sets the read-only status.

See [ListBoxClass Methods](#) for more information.

-> `list_item_for_display@`

Establishes the list item to be displayed in the list box

Class ListBoxClass set

Format `this.list_item_for_display@(itemNumber)`

Arguments itemNumber An integer. This index indicates the item to be displayed at the top of the list box.

Description Established the index of the element that will be displayed at the top of the list box. If the method `force_list_item_display@` is set to TRUE, the item specified by the `list_item_for_display@` method is shown at the top of the list box, if possible.

See also [ListBoxClass->force list item display@](#)

-> text_color@

Sets text color

Class ListBoxClass set

Format this.text_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method text_color@ sets the text color with an array of values. Text appears in the list box.

See [ListBoxClass Methods](#) for more information.

-> text_color_cmyk@

Sets text CMYK color

Class ListBoxClass set

Format this.text_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method text_color_cmyk@ sets the text color with CMYK values. Text appears in the list box.

See [ListBoxClass Methods](#) for more information.

-> text_color_name@

Sets text name color

Class ListBoxClass set

Format this.text_color_name@(name)

Arguments name The color name.

Description The set method text_color_name@ sets the text color by name. Text appears in the list box.

See [ListBoxClass Methods](#) for more information.

-> text_color_rgb@

Sets text RGB color

Class ListBoxClass set

Format this.text_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the list box.

See [ListBoxClass Methods](#) for more information.

-> `text_font_attrs@`

Sets text font information

Class ListBoxClass set

Format `this.text_font_attrs@(format font_attrs_info@ fonts)`

Arguments fonts The font information:
 font_name The font name.
 font_size The font point size.
 bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
 italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
 shadow Not used.

Description The set method `text_font_attrs@` sets all the text font information. Text appears in the list box. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [ListBoxClass Methods](#) for more information.

-> `text_font_bold@`

Sets text bold state

Class ListBoxClass set

Format `this.text_font_bold@(flag)`

Arguments flag Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.

Description The set method `text_font_bold@` sets the text bold state. Text appears in the list box. Use this method with the get method `text_font_bold@` to verify and change the object text bold state. Use the set method `text_font_attrs@` to set all font information in one step.

See [ListBoxClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class ListBoxClass set

Format `this.text_font_italic@(flag)`

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method `text_font_italic@` sets the italic state of the object text. Text appears in the list box. Use this method with the get method `text_font_italic@` to verify and change the object text italic state. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [ListBoxClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class ListBoxClass set

Format `this.text_font_name@(name)`

Arguments name A string for the font.

Description The set method `text_font_name@` returns the object text font name. Text appears in the list box. Use this method with the get method `text_font_name@` to verify and change the object text font. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [ListBoxClass Methods](#) for more information.

-> `text_font_size@`

Sets text font size

Class ListBoxClass set

Format `this.text_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `text_font_size@` sets the object text font size. Text appears in the list box. Use this method with the get method `text_font_size@` to verify and change the object text size. Use the set method `text_font_attrs@` to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [ListBoxClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class ListBoxClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method text_is_shadowed@ sets the shadow state of the object text. Use this method with the get method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the list box. Text only appears shadowed on color displays.

See [ListBoxClass Methods](#) for more information.

-> text_strings@

Sets list box contents

Class ListBoxClass set

Format this.text_strings@(valArray)

Arguments valArray An array of strings for the list box contents.

Description The set method text_strings@ sets the list box contents. Use this method with the get method text_strings@ to verify and change the list box contents.

For example, to set the list box contents use the following:

IF this.text_strings@ = NULL	'get method
this.text_strings@ = {"red", "white", "blue"}	'set method

See [ListBoxClass Methods](#) for more information.

-> thickness@

Sets list box thickness

Class ListBoxClass set

Format this.thickness@(pixels)

Arguments pixels The thickness, in pixels.

Description The set method thickness@ sets the list box thickness in pixels. The thickness is the bevel appearance of the list box sides.

See [ListBoxClass Methods](#) for more information.

-> value@

Sets current list box choice

Class ListBoxClass set

Format this.value@(value)

Arguments value A string or array of strings for the current list box choice.

Description The set method value@ sets the current list box choice. If is_multi_select@ is set to FALSE the method only accepts a string as an argument. If is_multi_select@ is set to TRUE the method accepts a string or an array of strings as an argument

See [ListBoxClass Methods](#) for more information.

-> value_index@

Sets current list box choice to array index

Class ListBoxClass set

Format this.value_index@(index)

Arguments index A numeric value or array of numeric values.

Description The set method `value_index@` sets the current list box choice to the array string index of the passed value.

For example, to set the index of the current list box choice to the array index of the string blue use the following:

```
var index, values
values = this.text_strings@
FOR index = 0 to ARRAY_SIZE@(values)-1
    IF values[index] = "blue"
        this.value_index@ = index
NEXT index
```

See [ListBoxClass Methods](#) for more information.

-> `vscroll_is_enabled@`

Sets vertical scroll bar status

Class ListBoxClass set

Format `this.vscroll_is_enabled@(flag)`

Arguments flag A Boolean value. TRUE makes the vertical scroll bar enabled and visible. FALSE makes the scroll bar disabled and hidden.

Description The set method `vscroll_is_enabled@` sets the vertical scroll bar status.

See [ListBoxClass Methods](#) for more information.

-> `vscroll_length@`

Sets vertical scroll bar length

Class ListBoxClass set

Format `this.vscroll_length@(pixels)`

Arguments pixels The length of the scroll bar, in pixels

Description The set method `vscroll_length@` sets the vertical scroll bar length.

See [ListBoxClass Methods](#) for more information.

-> **vscroll_origin@**

Sets vertical scroll bar origin

Class ListBoxClass set

Format this.vscroll_origin@(pixels)

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the list box.

Description The set method vscroll_origin@ sets the vertical scroll bar origin.

See [ListBoxClass Methods](#) for more information.

-> **width@**

Sets list box width

Class PanelClass set

Format this.width@(pixels)

Arguments chars The width, in characters, of the list box.

Description The set method width@ sets the list box's width. Use this method with the get method width to verify and change the list box's width.

For example, to make the list box width at least 25 characters use the following:

```
var chars
chars = this.width@           'get method
IF chars < 25
    this.width@ = 25         'set method
```

See [ListBoxClass Methods](#) for more information.

<- button3_menu_state_event

Sets menu state before pop up menu is displayed

Class ListBoxClass event

Format flag = this.button3_menu_state_event(function)

Arguments function The name of a pop up menu function.

Description The button3_menu_state_event is called by Applixware Builder before a pop up menu is displayed for a control. The event should be programmed to return either a Boolean value or NULL. The event should return TRUE if the menu item for the function should be active. The event should return FALSE if the menu item for the function should be grayed and inactive. The event should return NULL if the state of the menu item is unchanged.

For example, the following event sets the state of different menu items:

```
get button3_menu_state_event(function)
  case of function
  case "menu_1", "menu_2"
    return(NULL) ' No change in status
  case "menu_3"
    return(TRUE) ' Active menu item
  case "menu_4"
    return(FALSE) ' Inactive menu item
  endcase
endget
```

Use the [button3_menu_info@](#) method to set the pop up menu information.

NOTE: The Debugger cannot access break points set in the button3_menu_state_event. Do not use break points in the button3_menu_state_event while you are debugging an application.

See [ListBoxClass Methods](#) for more information.

<- error_event

Called for system errors

Class ListBoxClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ListBoxClass Methods](#) for more information.

-> changed_event

Called when list box choice changes

Class ListBoxClass event

Format this.changed_event(value)

Arguments value A number or one-element array containing a number. The number is the index of the selected item in the list box, indexes are 0 based. A one-element array is returned if the list box allows multiple selections.

Description The changed_event is called by Applixware Builder when the the list box choice changes. The event is passed the new list box choice as an integer or one-element integer.

The changed_event is called for the first selection of a multiple selection when using CTRL-Click (pressing CTRL and a left mouse button click). This is a user-defined event, place all actions you want performed in the event definition. The following is an example of a list box changed_event.

```
set changed_event(value)
```

```

var choice
IF IS_ARRAY@(value)    ' check if array for multi-select
    choice = value[0]
ELSE
    choice = value
IF choice = 1
{
    ' actions when list box choice is 1
}
ELSE ' value is another list box choice
{
    ' actions when list box choice is not 1
}

```

See [ListBoxClass Methods](#) for more information.

-> double_click_event

Called when double-click in list box

Class ListBoxClass event

Format this.double_click_event(value)

Arguments value A one-element array containing the index number of the list box item receiving the double-click

Description The double_click_event is called by Applixware Builder when a double-click action occurs in the list box. This is a user-defined event, place all actions you want performed in the event definition.

See [ListBoxClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class ListBoxClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the list box dimensions, color, and so on.

See [ListBoxClass Methods](#) for more information.

-> multi_select_event

Called when multiple list box lines selected

Class ListBoxClass event

Format this.multi_select_event(valArray)

Arguments valArray An array of item indices, in the selection order of the items.

Description The multi_select_event is called by Applixware Builder when the multiple list box lines are selected by pressing CTRL and a left mouse button click. The **changed_event** is called for the first selection in the list box, the multi_select_event is called for subsequent CTRL-Click selections. Use a multiple select to select non-contiguous list box lines. The event is passed the list box selection as an array of item indices. This is a user-defined event, place all actions you want performed in the event definition.

See [ListBoxClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class ListBoxClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The set event `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [ListBoxClass Methods](#) for more information.

-> `stroke_select_event`

Called when list box lines are selected while a mouse button is pressed

Class `ListBoxClass` event

Format `this.stroke_select_event(valArray)`

Arguments `valArray` An array of item indices.

Description The `stroke_select_event` is called by Applixware Builder when list box lines are selected while a mouse button is pressed. The event is called when the mouse button is released.

A SHIFT-Click (pressing SHIFT and a left mouse button click) key sequence generates a `stroke_select_event`, selecting list box items from the selected line to the line at the mouse cursor position. This is a user-defined event, place all actions you want performed in the event definition.

See [ListBoxClass Methods](#) for more information.

-> `terminate_event`

Called before closing or destroying dialog box

Class `ListBoxClass` event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ListBoxClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class ListBoxClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [ListBoxClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class ListBoxClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [ListBoxClass Methods](#) for more information.

AB_FILE_SAVE_AS_PROMPT@

Displays the Save As dialog box

Format format doc_format info = AB_FILE_SAVE_AS_PROMPT@([format doc_format docinfo[, suffix[, title[, app]]]])

Arguments	docinfo	An array of information in doc_format_ format. The format is defined in the ELF header file fileinf_.am.
	suffix	The file suffix to use as a search wildcard. The argument can also be a 2 element array. suffix[0] is the string that appears in the File Type combo box. suffix[1] is the string appended to the file name when the file is saved.
	title	The title to use in the Save As dialog box. The title Save As is used if no title is passed.
	app	An array of application specific information.
Description	The macro displays the Save As dialog box and returns the file information in doc_format format. This is the same dialog box used in Applixware Builder.	

AB_FOPEN_PROMPT@

Displays the Open dialog box

Format path = AB_FOPEN_PROMPT@([suffix[, dir[, wildcard[, title[, helpid[, extns[, cbData]]]]]])

Arguments	suffix	The file suffix, such as ".ab" for Applixware Builder files.
	dir	The full path name of the directory.
	wildcard	The search wildcard, such as *, ?, and so on.
	title	The title to use in the Open dialog box. The title Open is used if no title is passed.
	helpid	The help ID to pass to the Applixware Help system when the Help button is clicked in the dialog box.
	extns	A two-dimensional array of file extension information. extns[x][0] is the string that appears in the File Type combo box. extns[x][1] is the search wildcard used for the selected file type.
	cbData	A 3 element array. cbData[0] is the object containing a callback method. cbData[1] is the callback method name. The callback must be a set method. cbData[2] is the data that is passed as the third argument to the callback method. The file name and a Boolean value indicating the read-only status of the file are passed as the first two arguments to the callback method.

Description The macro displays the Open dialog box and returns the file name with full path name. This is the same dialog box used in Applixware Builder.

See the [example](#) for information about using the extns and cbData arguments.

DROP_OBJECT_SERVER@

Closes axnet connection

Format DROP_OBJECT_SERVER@(host)

Arguments host The name of the host machine.

Description The macro closes an axnet connection to the host machine.

IS_OBJECT@

Indicates if passed variable is an object

Format flag = IS_OBJECT@(thing)

Arguments thing An ELF datum.

Description The macro returns a Boolean value indicating if the passed variable is an object. The macro returns TRUE if it is an object, otherwise it returns FALSE.

OBJECT_CREATE@

Creates an object

Format var object obj = OBJECT_CREATE@(className, objectName)
var object obj = OBJECT_CREATE@(classObjRef, objectName)

Arguments class The name of an object class in the current application from which the object inherits attributes and methods. An object class can be a standard Applixware Builder class, or a user-defined class.

className A string containing the name of a class within the Builder application. This name should be an Applixware Built-in class. To create an object from a user-defined class, use a class object reference, as described in the following paragraphs.

objReference A class object reference, as returned by the class_library@ method. This argument can also be a string containing the name of a class, but this is not recommended when creating an object from a user-defined class.

Description The macro creates and returns an object. You must still assign object attributes, such as parents and children, using BaseClass and inherited class methods. An object remains part of an application until it is removed with the OBJECT_DESTROY@ macro.

If you create an object from an Applixware built-in class, you can specify the class name for the first argument.

If you create an object from a user-defined class, you should use the `class_library@` method of the Application class to create a class object reference. Then, you can then pass this class object reference to the `OBJECT_CREATE@` macro to instantiate an object from that class. For example:

```
var object foo
```

```
foo = object_create@(this.application@.class_library@("MyUserDefinedClass"), "myObject")
```

In this example, the file `MyUserDefinedClass.cll` must have been loaded into Builder by selecting **Resources @ Load Classes**.

OBJECT_DESTROY@

Removes an object and all children

Format `OBJECT_DESTROY@(format object obj)`

Arguments `obj` The object to delete.

Description The macro removes the passed object and all the object's children from an application

OBJECT_EXISTS@

Returns object existence

Format `flag = OBJECT_EXISTS@(format object obj)`

Arguments `obj` An object

Description The macro returns a Boolean value indicating the object's existence. The macro returns TRUE if the object exists, otherwise it returns FALSE.

REMOTE_OBJECT_CREATE@

Creates a remote object

NOTE: This is an obsolete macro, use the ApplicationClass get method [**remote object create@**](#) to create remote objects in Applixware Builder applications.

Format format object obj = REMOTE_OBJECT_CREATE@(host, class, name)

Arguments

host	The name of the host machine where the object exists.
class	The name of an object class in the current application from which the object inherits attributes and methods. An object class can be a standard Applixware Builder class, or a user-defined class.
name	The object name, as a text string.

Description The macro creates and returns a remote object. An axnet process must exist on the host machine before you create a remote object with this macro. A remote object is referenced by the returned object handle, the object cannot be assigned as the child of a local application object. A remote object exists until it is removed with the REMOTE_OBJECT_DESTROY@ macro.

The axnet process on the host machine must be started using the full path name. If axnet is started without the full path name, or with a linked path name, the macro is unable to find the process and create a remote object.

REMOTE_OBJECT_DESTROY@

Removes a remote object

NOTE: This is an obsolete macro, use the ApplicationClass set method [remote object destroy@](#) to remove remote objects in Applixware Builder applications.

Format REMOTE_OBJECT_DESTROY@(host, format object obj)

Arguments

host	The name of the host machine where the object exists.
obj	The object to delete.

Description The macro removes the remote object from the host machine. The macro does not close the axnet connection to the remote host. Use the remote_object_server_close@ method in the CommonDlgClass to close the axnet connection.

REMOTE_OBJECT_FIND@

Returns handle to remote object

NOTE: This is an obsolete macro, use the ApplicationClass get method [remote object find@](#) to find remote objects in Applixware Builder applications.

Format format object obj = REMOTE_OBJECT_FIND@(host, name)

Arguments host The name of the host machine where the object exists.
name The name of the object to find.

Description The macro returns a handle to a remote object. The macro returns NULL if the the object does not exist on the remote host.

AB_FOPEN_PROMPT@ Example

In the following example the macro is called in the clicked_event of a ButtonClass object.

```
set clicked_event
  var extns, cbData, path
  /* set extension info */
  extns[0][0] = "Applixware Builder Files [* .ab]" ' String displayed in the combo box
  extns[0][1] = "*.ab" ' Used in the search
  extns[1][0] = "Applixware Builder Files [* .ab]"
  extns[1][1] = " *.abo"
  /* set callback info
  cbData[0] = this ' object to receive the callback
  cbData[1] = "file_open_callback" ' callback method
  cbData[2] = current_dir@()' this is the dir before the open_box was called.
  path = ab_fopen_prompt@(".ab",current_dir@(), "*.ab","MyOpen", "MyHelp",extns,cbData)
...
endset
```

The following method is the callback method called when opening a file in the clicked_event.

```
set file_open_callback(filename, readOnly, origDir)
  var boo
  if origDir <> user_dir@() {
    beep@()
    info_message@("Sorry, you can only open files in your user directory.")
    return
  }
  boo = read_data_file@(filename)
...
endset
```

Return to [AB_FOPEN_PROMPT@](#) .

<- alternate_reply_recipient@

Returns mail message alternate reply recipient

Class MailClass get

Format recipient = this.alternate_reply_recipient@

Description The get method `alternate_reply_recipient@` returns the alternate reply recipient of the mail message. The alternate reply recipient is the person who will receive replies to your message.

See [MailClass Methods](#) for more information.

<- attachments@

Returns mail message attachments

Class MailClass get

Format fileArray = this.attachments@

Description The get method `attachments@` returns the attached files to the mail message as an array.

See [MailClass Methods](#) for more information.

<- bc_recips@

Returns blind copy recipients

Class MailClass get

Format recipArray = this.bc_recips@

Description The get method `bc_recips@` returns the blind copy recipients of the mail message as an array of strings.

See [MailClass Methods](#) for more information.

<- body@

Returns message body

Class MailClass get

Format msgArray = this.body@

Description The get method body@ returns the message body as an array of strings.

See [MailClass Methods](#) for more information.

<- cc_recips@

Returns carbon copy recipients

Class MailClass get

Format recipArray = this.cc_recips@

Description The get method cc_recips@ returns the carbon copy recipients of the mail message as an array of strings.

See [MailClass Methods](#) for more information.

<- is_certified@

Returns certified status

Class MailClass get

Format flag = this.is_certified@

Description The get method is_certified@ returns the certified status as a Boolean value. The method returns TRUE if the message is certified, otherwise it returns FALSE.

See [MailClass Methods](#) for more information.

<- is_interactive@

Returns interactive status

Class MailClass get

Format flag = this.is_interactive@

Description The get method `is_interactive@` returns the interactive status as a Boolean value. The method returns TRUE if the MailClass object is in interactive mode, otherwise it returns FALSE.

If the object is in interactive mode, the Send Mail dialog box appears before the message is sent, loaded with the current message attributes. If the object is not in interactive mode, the mail message is sent silently, provided the mail message has at least one recipient and a subject.

See [MailClass Methods](#) for more information.

<- is_reply_requested@

Returns reply request status

Class MailClass get

Format flag = this.is_reply_requested@

Description The get method `is_reply_requested@` returns the reply request status as a Boolean value. The method returns TRUE if a reply request is made, otherwise it returns FALSE.

See [MailClass Methods](#) for more information.

<- is_urgent@

Returns urgent status

Class MailClass get

Format flag = this.is_urgent@

Description The get method `is_urgent@` returns the message urgent status as a Boolean value. The method returns TRUE if the message is tagged as urgent, otherwise it returns FALSE.

See [MailClass Methods](#) for more information.

<- outbox_copy@

Returns copy to outbox status

Class MailClass get

Format flag = this.outbox_copy@

Description The get method `reply_requested@` returns the copy to outbox status as a Boolean value. The method returns TRUE if the message is copied to your outbox when the message is sent, otherwise it returns FALSE.

See [MailClass Methods](#) for more information.

<- reply_text@

Returns reply text

Class MailClass get

Format text = this.reply_text@

Description The get method `reply_text@` returns the reply text as a string.

See [MailClass Methods](#) for more information.

<- subject@

Returns message subject

Class MailClass get

Format text = this.subject@

Description The get method `subject@` returns the message subject.

See [MailClass Methods](#) for more information.

<- to_recips@

Returns message recipients

Class MailClass get

Format recipArray = this.to_recips@

Description The get method to_recips@ returns the recipients of the mail message as an array of strings.

See [MailClass Methods](#) for more information.

-> alternate_reply_recipient@

Sets mail message alternate reply recipient

Class MailClass set

Format this.alternate_reply_recipient@(recipient)

Arguments recipient The e-mail address of the alternate reply recipient.

Description The set method alternate_reply_recipient@ sets the alternate reply recipient of the mail message. The alternate reply recipient is the person who will receive replies to your message.

See [MailClass Methods](#) for more information.

-> attachments@

Sets message attachments

Class MailClass set

Format this.attachments@(fArray)

Arguments fArray An array of files.

Description The set method attachments@ sets the message attachments.

See [MailClass Methods](#) for more information.

-> **bc_recips@**

Sets message blind copy recipients

Class MailClass set

Format this.bc_recips@(rec)

Arguments rec A string or array of strings.

Description The set method bc_recips@ sets the message blind copy recipients.

See [MailClass Methods](#) for more information.

-> **body@**

Sets message body

Class MailClass set

Format this.body@(msg)

Arguments msg A string or array of strings.

Description The set method body@ sets the message body.

See [MailClass Methods](#) for more information.

-> **cc_recips@**

Sets message carbon copy recipients

Class MailClass set

Format this.cc_recips@(rec)

Arguments rec A string or array of strings.

Description The set method cc_recips@ sets the message carbon copy recipients.

See [MailClass Methods](#) for more information.

-> defaults@

Sets message defaults

Class MailClass set

Format this.defaults@

Description The set method defaults@ sets the message defaults. All message attributes are cleared.

See [MailClass Methods](#) for more information.

-> is_certified@

Sets message certified status

Class MailClass set

Format this.is_certified@(flag)

Arguments flag A Boolean value. TRUE enables the message certified status, FALSE disables the message certified status.

Description The set method certified@ sets the message certified status.

See [MailClass Methods](#) for more information.

-> is_interactive@

Sets message interactive mode

Class MailClass set

Format this.is_interactive@(flag)

Arguments flag A Boolean value. TRUE enables interactive mode, FALSE disables interactive mode.

Description The set method is_interactive@ sets the message interactive mode.

If the object is in interactive mode, the Send Mail dialog box appears before the message is sent, loaded with the current message attributes. If the object is not in interactive mode, the mail message is sent silently, provided the mail message has at least one recipient and a subject.

See [MailClass Methods](#) for more information.

-> is_reply_requested@

Sets reply request status

Class MailClass set

Format this.is_reply_requested@(flag)

Arguments flag A Boolean value. TRUE sets the reply request, FALSE disables the reply request.

Description The set method is_reply_requested@ sets the reply request status.

See [MailClass Methods](#) for more information.

-> is_urgent@

Sets message urgent status

Class MailClass set

Format this.is_urgent@(flag)

Arguments flag A Boolean value. TRUE sets the urgent status, FALSE disables the urgent status.

Description The set method is_urgent@ sets the message urgent status.

See [MailClass Methods](#) for more information.

-> outbox_copy@

Sets copy to outbox status

Class MailClass set

Format this.outbox_copy@(flag)

Arguments flag A Boolean value. TRUE copies the message to your outbox when it is sent.

Description The set method certified@ sets the copy to outbox status.

See [MailClass Methods](#) for more information.

-> reply_text@

Sets reply request text

Class MailClass set

Format this.reply_text@(text)

Arguments text A string.

Description The set method reply_text@ sets the reply request text. The reply request text appears in the message when reply_requested@ is TRUE.

See [MailClass Methods](#) for more information.

-> reset@

Sets message defaults

Class MailClass set

Format this.reset@

Description The set method reset@ sets the message defaults. All message attributes are cleared.

See [MailClass Methods](#) for more information.

-> send_mail@

Sends a mail message

Class MailClass set

Format this.send_mail@

Description The set method send_mail@ sends a mail message. If interactive@ is set to TRUE, the mail message appears in the Send Mail dialog box before it is sent. If interactive@ is set to FALSE the message is sent immediately without user feedback. The Send Mail dialog box appears if the message recipients or subject are not set, even if interactive@ is set to FALSE.

See [MailClass Methods](#) for more information.

-> subject@

Sets message subject

Class MailClass set

Format this.subject@(text)

Arguments text A string.

Description The set method subject@ sets the message subject.

See [MailClass Methods](#) for more information.

-> to_recips@

Sets message recipients

Class MailClass set

Format this.to_recips@(rec)

Arguments rec A string or array of strings.

Description The set method to_recips@ sets the message recipients.

See [MailClass Methods](#) for more information.

<- error_event

Called for system errors

Class MailClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [MailClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class MailClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [MailClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class MailClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [MailClass Methods](#) for more information.

-> `time_out_event`

Called on object timer time-out

Class MailClass event

Format `this.time_out_event`

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [MailClass Methods](#) for more information.

<- `data@`

Returns the menu bar information as an ELF array

Class MenuBarClass get

Format `infoArray = this.data@`

Description The get method `data@` returns the menu bar information as an ELF array. You can create your own menu bar using an array, the set method `data@`, and the `READ_DATA_FILE@` and `WRITE_DATA_FILE@` ELF macros.

See [MenuBarClass Methods](#) for more information.

-> data@

Sets the menu bar information

Class MenuBarClass set

Format this.data@(infoArray)

Arguments infoArray An ELF array of menu bar information. The array contains menu item names, submenus, and macros or methods called by a menu choice.

Description The set method dat@a sets the menu bar information from a passed ELF array. You can create your own menu bar using an array, the set method data@, and the READ_DATA_FILE@ and WRITE_DATA_FILE@ ELF macros.

See [MenuBarClass Methods](#) for more information.

-> dbox_close@

Closes a dialog box

Class MenuBarClass set

Format this.dbox_close@(name)

Arguments name The dialog box to close. If no argument is, the method closes the parent dialog box containing the menu bar.

Description The set method dbox_close@ closes a dialog box.

See [MenuBarClass Methods](#) for more information.

-> dbox_open@

Opens a dialog box

Class MenuBarClass set

Format this.dbox_open@(name)

Arguments name The dialog box to open.

Description The set method `dbox_open@` opens a dialog box.

See [MenuBarClass Methods](#) for more information.

-> menu_item_name@

Sets display string of a menu item

Class MenuBarClass set

Format `this.menu_item_name@(function, name)`

Arguments

<code>function</code>	The name of a menu function.
<code>name</code>	The string to display in the menu bar for the function.

Description The set method `menu_item_name@` sets the display string of a menu bar item. Use this method when you need to display dynamic strings for a menu item, such as an undo feature.

See [MenuBarClass Methods](#) for more information.

-> menu_item_status@

Sets display status of a menu item

Class MenuBarClass set

Format `this.menu_item_status@(function, value)`

Arguments

<code>function</code>	The name of a menu function.
<code>value</code>	The status of the menu bar item. The menu bar status values are defined in the <code>/install_dir/axdata/elf/menubar_.am</code> header file. include the header file in the method source where you make calls to this method. The status values are defined as follows:

<code>MENUSTAT#NORMAL</code>	Display normally
<code>MENUSTAT#DIMMED</code>	Display grayed
<code>MENUSTAT#TOGGLE_ON</code>	Toggle on to left
<code>MENUSTAT#TOGGLE_OFF</code>	Is a toggle, but off now
<code>MENUSTAT#RADIO_OFF</code>	Is a radio, currently off
<code>MENUSTAT#RADIO_ON</code>	Radio on to left

MENUSTAT#NO_SHOW Suppress display altogether

Description The set method `menu_item_name@` sets the display status of a menu bar item. Use the method to set the status of a menu bar choice, making it available, grayed, or even a toggle or radio button.

See [MenuBarClass Methods](#) for more information.

<- error_event

Called for system errors

Class MenuBarClass event

Format `flag = this.error_event(error_object)`

Arguments `error_object` An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [MenuBarClass](#) methods to get error object information.

See [MenuBarClass Methods](#) for more information.

<- menu_name_event

Sets menu item names

Class MenuBarClass event

Format `flag = this.menu_name_event(function)`

Arguments `function` The name of a menu function.

Description The `menu_name_event` is called by Applixware Builder before a menu is displayed to set the menu item names. The event should be programmed to return a string. Use

this event to create dynamic menu bar items with names that change to reflect different states in your application

For example, the following event sets the name of different menu items:

```
get menu_name_event(function)
  case of function
  case "menu_1"
    return("Orange")
  case "menu_3"
    return("Apple")
  case "menu_4"
    return("Pear")
  endcase
endget
```

See [MenuBarClass Methods](#) for more information.

<- menu_pick_event

Called when a menu choice changes

Class MenuBarClass event

Format flag = this.menu_pick_event(value)

Arguments value The method or macro associated with the selected menu item.

Description The menu_pick_event is called by Applixware Builder when a menu choice changes. The Applixware Builder, by default, executes the macro or method associated with a menu choice. You can program this event to handle the menu bar functionality. The event should return TRUE if you handle the function calls, otherwise it should return FALSE. This is a user-defined event, place all actions you want performed in the event definition. For example, the default menu bar defines menu_pick_event as follows:

```
get menu_pick_event(val)
  var pos, func, arg
  pos = string_index@(val, " ")
  if pos > 0 {
    func = trim@(substring@(val,1,pos-1))
```

```

        arg = trim@(substring@(val,pos+1))
        this.*!func(arg)
    }
    else
        this.*!val
    return(TRUE)
endset

```

See [MenuBarClass Methods](#) for more information.

<- menu_state_event

Sets menu state before menu is displayed

Class MenuBarClass event

Format flag = this.menu_state_event(function)

Arguments function The name of a menu function.

Description The menu_state_event is called by Applixware Builder before a menu is displayed. The event should be programmed to return a display value defined in the header file `menubar_.am`, located in the `install_dir/axdata/elf` directory. The defined display values are:

MENUSTAT#NORMAL	Display normally
MENUSTAT#DIMMED	Display grayed
MENUSTAT#TOGGLE_ON	Toggle on to left
MENUSTAT#TOGGLE_OFF	Is a toggle, but off now
MENUSTAT#RADIO_OFF	Is a radio, currently off
MENUSTAT#RADIO_ON	Radio on to left
MENUSTAT#NO_SHOW	Suppress display altogether

For example, the following event sets the state of different menu items:

```

#include "menubar_.am"
get menu_state_event(function)
    case of function
        case "menu_1", "menu_2"

```

```
        return(MENUSTAT#NORMAL) ' Display normal
    case "menu_3"
        return(MENUSTAT#DIMMED) ' Display grayed
```

NOTE: The Debugger cannot access break points set in the menu_state_event. Do not use break points in the menu_state_event while you are debugging an application.

See [MenuBarClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class MenuBarClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [MenuBarClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class MenuBarClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [MenuBarClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class MenuBarClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [MenuBarClass Methods](#) for more information.

-> update_event

Called to update dialog box widget

Class MenuBarClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box widget. This is a user-defined event, place all actions you want performed in the event definition.

See [MenuBarClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class OptionMenuClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the option menu. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

1 RGB

- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [OptionMenuClass Methods](#) for more information.

<- is_drop_shadowed@

Returns option menu shadowed state

Class OptionMenuClass get

Format flag = this.is_drop_shadowed@

Description The get method is_drop_shadowed@ returns a Boolean value indicating the option menu shadowed state. If the method returns TRUE the option menu is shadowed, otherwise the method returns FALSE. A drop shadow gives a option menu a 3-dimensional appearance.

See [OptionMenuClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class OptionMenuClass get

Format colors = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears in the option menu.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [OptionMenuClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class OptionMenuClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the option menu. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [OptionMenuClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class OptionMenuClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. You can use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the option menu.

For example, to make the object text bold use the following:

```
var flag
```

```
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [OptionMenuClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class OptionMenuClass get

Format flag = this.text_font_italic@

Description The get method `text_font_italic@` returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method `text_font_italic@` to verify and change the object text italic state.

Text appears in the option menu.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@         'get method
IF NOT flag
    this.text_font_italic@ = TRUE     'set method
```

See [OptionMenuClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class OptionMenuClass get

Format name = this.text_font_name@

Description The get method `text_font_name@` returns the object text font name. You can use this method with the set method `text_font_name@` to verify and change the object text font.

Text appears in the option menu.

For example, to set the object text font to Courier use the following:

```

var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"   'set method

```

See [OptionMenuClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class OptionMenuClass get

Format size = this.text_font_size@

Description The get method `text_font_size@` returns the object text font size. You can use this method with the set method `text_font_size@` to verify and change the object text size.

Text appears in the option menu.

For example, to set the object text size to 12 use the following:

```

var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method

```

See [OptionMenuClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class OptionMenuClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the option menu. Text only appears shadowed on color displays.

See [OptionMenuClass Methods](#) for more information.

<- text_strings@

Returns option menu choices

Class OptionMenuClass get

Format valArray = this.text_strings@

Description The get method text_strings@ returns the option menu choices as an array of strings. You can use this method with the set method text_strings@ to verify and change the option menu choices.

For example, to set the option menu choices you would use the method as follows:

```
IF this.text_strings@ = NULL           'get method
    this.text_strings@ = {"red", "white", "blue"} 'set method
```

See [OptionMenuClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class OptionMenuClass get

Format colorArray = this.title_color@

Description The get method title_color@ returns a two dimensional array of title color settings.

Title is the text that appears adjacent to the option menu.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [OptionMenuClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class OptionMenuClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name The font name.

font_size The font point size.

bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

shadow Not used.

See [OptionMenuClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class OptionMenuClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears adjacent to the option menu.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [OptionMenuClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class OptionMenuClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears adjacent to the option menu.

For example, to make the object title italic use the following:

```
var flag
```

```
flag = this.title_font_italic@           'get method
IF NOT flag
    this.title_font_italic@ = TRUE       'set method
```

See [OptionMenuClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class OptionMenuClass get

Format name = this.title_font_name@

Description The get method `title_font_name@` returns the object title font name. You can use this method with the set method `title_font_name@` to verify and change the object text font.

Title is the text that appears adjacent to the option menu.

For example, to set the object title font to Courier use the following:

```
var name
name = this.title_font_name@           'get method
IF name <> "Courier"
    this.title_font_name@ = "Courier"  'set method
```

See [OptionMenuClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class OptionMenuClass get

Format size = this.title_font_size@

Description The get method `title_font_size@` returns the object title font size. You can use this method with the set method `title_font_size@` to verify and change the object text size.

Title is the text that appears adjacent to the option menu.

For example, to set the object title size to 12 use the following:

```
var size
```

```
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [OptionMenuClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class OptionMenuClass get

Format flag = this.title_is_shadowed@

Description The get method `title_is_shadowed@` returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. Use this method with the set method `title_is_shadowed@` to verify and change the object text shadow state.

Title is the text that appears adjacent to the option menu. Text only appears shadowed on color displays.

See [OptionMenuClass Methods](#) for more information.

<- value@

Returns current option menu choice

Class OptionMenuClass get

Format value = this.value@

Description The get method `value@` returns the current option menu choice as a string. You can use this method with the set method `value@` to verify and change the option menu choices.

For example, to set the current option menu choice you would use the method as follows:

```
IF this.value@ <> "blue"               'get method
    this.value@ = "blue"                 'set method
```

See [OptionMenuClass Methods](#) for more information.

<- value_index@

Returns array index of option menu choice

Class OptionMenuClass get

Format index = this.value_index@

Description The get method value_index@ returns the array index of the current option menu choice as a numeric value. The option menu choices are a 0-based text string array.

For example, to get the index of the current option menu choice you would use the method as follows:

```
var values, index
values = this.text_strings@
index = this.value_index@
IF values[index] <> "blue"
    this.value@ = "blue"
```

See [OptionMenuClass Methods](#) for more information.

-> control_color@

Sets control color

Class OptionMenuClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB

6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values. See [OptionMenuClass Methods](#) for more information.

-> `control_color_cmyk@`

Sets control CMYK color

Class OptionMenuClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [OptionMenuClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class OptionMenuClass set

Format this.control_color_is_workspace@(flag)

Arguments flag Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.

Description The set method control_color_is_workspace@ sets the color used by the object. See [OptionMenuClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class OptionMenuClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name. See [OptionMenuClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class OptionMenuClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values. See [OptionMenuClass Methods](#) for more information.

-> `display@`

Displays option menu

Class OptionMenuClass set

Format `this.display@`

Description The set method `display@` displays the option menu in the dialog box. See [OptionMenuClass Methods](#) for more information.

-> `is_drop_shadowed@`

Sets option menu shadowed state

Class OptionMenuClass set

Format `this.is_drop_shadowed@(flag)`

Arguments `flag` A Boolean value. TRUE makes the option menu the drop shadowed.

Description The set method `is_drop_shadowed@` sets the option menu shadowed state. A drop shadow gives a option menu a 3-dimensional appearance.

See [OptionMenuClass Methods](#) for more information.

-> `text_color@`

Sets text color

Class OptionMenuClass set

Format `this.text_color@(type, c1, c2, c3, c4)`

Arguments `type` The color type. The valid color types are:
1 RGB

- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears in the option menu.

See [OptionMenuClass Methods](#) for more information.

-> `text_color_cmyk@`

Sets text CMYK color

Class OptionMenuClass set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.

black The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears in the option menu.

See [OptionMenuClass Methods](#) for more information.

-> `text_color_name@`

Sets text name color

Class OptionMenuClass set

Format `this.text_color_name@(name)`

Arguments name The color name.

Description The set method `text_color_name@` sets the text color by name. Text appears in the option menu.

See [OptionMenuClass Methods](#) for more information.

-> `text_color_rgb@`

Sets control RGB color

Class OptionMenuClass set

Format `this.text_color_rgb@(red, green, blue)`

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the option menu.

See [OptionMenuClass Methods](#) for more information.

-> text_font_attrs@

Sets text font information

Class OptionMenuClass set

Format this.text_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method text_font_attrs@ sets all the text font information. Text appears in the option menu. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format.

See [OptionMenuClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class OptionMenuClass set

Format this.text_font_bold@(flag)

Arguments

flag	Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.
------	---

Description The set method text_font_bold@ sets the text bold state. Text appears in the option menu. You can use this method with the get method text_font_bold@ to verify and change the object text bold state. Use the set method text_font_attrs@ to set all font information in one step.

See [OptionMenuClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class OptionMenuClass set

Format this.text_font_italic@(flag)

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method text_font_italic@ sets the italic state of the object text. Text appears in the option menu. You can use this method with the get method text_font_italic@ to verify and change the object text italic state. Use the set method text_font_attrs@ to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [OptionMenuClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class OptionMenuClass set

Format this.text_font_name@(name)

Arguments name A string for the font.

Description The set method text_font_name@ returns the object text font name. Text appears in the option menu. You can use this method with the get method text_font_name@ to verify and change the object text font. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
```

```
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [OptionMenuClass Methods](#) for more information.

-> text_font_shadow@

Sets text shadow type

Class OptionMenuClass set

Format this.text_font_shadow@(type)

Arguments type The shadow type. The defined shadow types are:

Description The set method text_font_shadow@ sets the object text shadow type. An object text is shadowed when the [text is shadowed@](#) method is set to TRUE.

Text appears in the option menu. Text only appears shadowed on color displays.

See [OptionMenuClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class OptionMenuClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears in the option menu. You can use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
```

IF size <> 12

 this.text_font_size@ = 12

'set method

See [OptionMenuClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class OptionMenuClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method text_is_shadowed@ sets the shadow state of the object text. Use this method with the get method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the option menu. Text only appears shadowed on color displays.

See [OptionMenuClass Methods](#) for more information.

-> text_strings@

Sets option menu choices

Class OptionMenuClass set

Format this.text_strings@(valArray)

Arguments valArray An array of strings for the option menu choices.

Description The set method text_strings@ sets the option menu choices. You can use this method with the get method text_strings@ to verify and change the option menu choices.

For example, to set the option menu choices you would use the method as follows:

IF this.text_strings@ = NULL

'get method

 this.text_strings@ = {"red", "white", "blue"}

'set method

See [OptionMenuClass Methods](#) for more information.

-> title_color@

Sets title color

Class OptionMenuClass set

Format this.title_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method title_color@ sets the title color with an array of values.

See [OptionMenuClass Methods](#) for more information.

-> title_color_cmyk@

Sets title CMYK color

Class OptionMenuClass set

Format this.title_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method title_color_cmyk@ sets the title color with CMYK values.
See [OptionMenuClass Methods](#) for more information.

-> title_color_name@

Sets title name color

Class OptionMenuClass set

Format this.title_color_name@(name)

Arguments name The color name.

Description The set method title_color_name@ sets the title color by name.
See [OptionMenuClass Methods](#) for more information.

-> title_color_rgb@

Sets control RGB color

Class OptionMenuClass set

Format this.title_color_rgb@(red, green, blue)

Arguments red The RGB red value.

green The RGB blue value.
blue The RGB green value.

Description The set method `title_color_rgb@` sets the title color with RGB values.
See [OptionMenuClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class OptionMenuClass set

Format `this.title_font_attrs@(format font_attrs_info@ fonts)`

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `title_font_attrs@` sets all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [OptionMenuClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class OptionMenuClass set

Format `this.title_font_bold@(flag)`

Arguments

flag	Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.
------	--

Description The set method `title_font_bold@` sets the title bold state. You can use this method with the get method `title_font_bold@` to verify and change the object title bold state. Use the set method `title_font_attrs@` to set all font information in one step.

See [OptionMenuClass Methods](#) for more information.

-> `title_font_italic@`

Sets title italic state

Class OptionMenuClass set

Format `this.title_font_italic@(flag)`

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method `title_font_italic@` sets the italic state of the object title. You can use this method with the get method `title_font_italic@` to verify and change the object title italic state. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [OptionMenuClass Methods](#) for more information.

-> `title_font_name@`

Sets title font name

Class OptionMenuClass set

Format `this.title_font_name@(name)`

Arguments name A string for the font.

Description The set method `title_font_name@` returns the object title font name. You can use this method with the get method `title_font_name@` to verify and change the object title font. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [OptionMenuClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class OptionMenuClass set

Format this.title_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method title_font_size@ sets the object title font size. You can use this method with the get method title_font_size@ to verify and change the object title size. Use the set method title_font_attrs@ to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@         'get method
IF size <> 12
    this.title_font_size@ = 12      'set method
```

See [OptionMenuClass Methods](#) for more information.

-> title_is_shadowed@

Sets title shadow state

Class OptionMenuClass set

Format this.title_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method `title_is_shadowed@` sets the shadow state of the object title. Use this method with the get method `title_is_shadowed@` to verify and change the object title shadow state.

Title is the text that appears adjacent to the option menu. Text only appears shadowed on color displays.

See [OptionMenuClass Methods](#) for more information.

-> `value@`

Sets current option menu choice

Class OptionMenuClass set

Format `this.value@(value)`

Arguments value A string for the current option menu choice.

Description The set method `value@` sets the current option menu choice.

For example, to set the current option menu choice you would use the method as follows:

IF <code>this.value@ <> "blue"</code>	'get method
<code>this.value@ = "blue"</code>	'set method

See [OptionMenuClass Methods](#) for more information.

-> `value_index@`

Sets current option menu choice to array index

Class OptionMenuClass set

Format `this.value_index@(index)`

Arguments index A numeric value.

Description The set method `value_index@` sets the current option menu choice to the array string index of the passed value. The option menu choices are a 0-based text string array.

For example, to set the index of the current option menu choice to the array index of the string blue you would use the method as follows:

```
var index, values
```

```

values = this.text_strings@
FOR index = 0 to ARRAY_SIZE@(values)-1
    IF values[index] = "blue"
        this.value_index@ = index
    NEXT index

```

See [OptionMenuClass Methods](#) for more information.

-> **changed_event**

Called when option menu choice changes

Class OptionMenuClass event

Format this.changed_event(index)

Arguments index A numeric value indicating the new menu choice index position. The option menu choices are a 0-based text string array.

Description The set event changed is called by Applixware Builder when the the option menu choice changes. This is a user-defined event, place all actions you want performed in the event definition. The following is an example of a option menu changed event.

```

set changed_event(index)
    IF value = "blue"
    {
        ' actions when option menu choice is blue
    }
    ELSE ' value is another option menu choice
    {
        ' actions when option menu choice is not blue
    }
endset

```

See [OptionMenuClass Methods](#) for more information.

<- error_event

Called for system errors

Class OptionMenuClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [OptionMenuClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class OptionMenuClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the option menu color, position, and so on.

See [OptionMenuClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class OptionMenuClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The set event `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [OptionMenuClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class OptionMenuClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [OptionMenuClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class OptionMenuClass event

Format this.time_out_event

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [OptionMenuClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class OptionMenuClass event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [OptionMenuClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class PanelClass get

Format `colorArray = this.control_color@`

Description The get method `control_color@` returns the color used by the panel. The array information is returned as follows:

`colorArray[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [PanelClass Methods](#) for more information.

<- height@

Returns panel height

Class PanelClass get

Format pixels = this.height@

Description The get method height@ returns the panel height in pixels. You can use this method with the set method height@ to verify and change the object's height.

For example, to make the panel height at least 10 pixels you would use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 10
    this.height@ = 10         'set method
```

See [PanelClass Methods](#) for more information.

<- style@

Returns panel style

Class PanelClass get

Format style = this.style@

Description The get method `style@` returns the panel style as an integer. A panel can be either plain or labeled. A labeled panel has its title displayed on the top edge of the panel. The method returns NULL if the value is not set. The values for panel styles, as defined in the *install_dir/axdata/elf/dialog_.am* include file are:

- | | | |
|---|-------------|---------------|
| 1 | RECT#PLAIN | Plain panel |
| 2 | RECT#LBELED | Labeled panel |

See [PanelClass Methods](#) for more information.

<- thickness@

Returns panel thickness

Class PanelClass get

Format pixels = this.thickness@

Description The get method `thickness@` returns the panel thickness in pixels. The thickness is the bevel appearance of the panel sides.

See [PanelClass Methods](#) for more information.

<- width@

Returns panel width

Class PanelClass get

Format pixels = this.width@

Description The get method `width@` returns the panel width in pixels. You can use this method with the set method `width@` to verify and change the object's width.

For example, to make the panel width at least 25 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 25
    this.width@ = 25          'set method
```

See [PanelClass Methods](#) for more information.

-> control_color@

Sets control color

Class PanelClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.

- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values. See [PanelClass Methods](#) for more information.

-> `control_color_cmyk@`

Sets control CMYK color

Class PanelClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [PanelClass Methods](#) for more information.

-> `control_color_is_workspace@`

Sets color used by object

Class PanelClass set

Format `this.control_color_is_workspace@(flag)`

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method `control_color_is_workspace@` sets the color used by the object. See [PanelClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class PanelClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name.

See [PanelClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class PanelClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the control color with RGB values.

See [PanelClass Methods](#) for more information.

-> height@

Sets panel height

Class PanelClass set

Format this.height@(value)

Arguments value The height, in pixels, of the panel.

Description The set method `height@` sets the panels's height. You can use this method with the get method `height@` to verify and change the object's height.

For example, to make the panel height at least 10 pixels you would use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 10
    this.height@ = 10         'set method
```

See [PanelClass Methods](#) for more information.

-> `style@`

Sets panel style

Class PanelClass get

Format `this.style@(type)`

Arguments `type` The panel type. The values for panel styles, as defined in the `install_dir/axdata/elf/dialog_.am` include file are:

RECT#PLAIN Plain panel
RECT#LBELED Labeled panel

Description The set method `style@` sets the panel style to the passed style type. A panel can be either plain or labeled. A labeled panel has its title displayed on the top edge of the panel.

See [PanelClass Methods](#) for more information.

-> `thickness@`

Sets panel thickness

Class PanelClass set

Format `this.thickness@(pixels)`

Arguments `pixels` The thickness, in pixels.

Description The set method `thickness@` sets the panel thickness in pixels. The thickness is the bevel appearance of the panel sides.

See [PanelClass Methods](#) for more information.

-> width@

Sets panel width

Class PanelClass set

Format `this.width@(value)`

Arguments value The width, in pixels, of the panel.

Description The set method `width@` sets the panels's width. You can use this method with the get method `width@` to verify and change the object's width.

For example, to make the panel width at least 25 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 25
    this.width@ = 25         'set method
```

See [PanelClass Methods](#) for more information.

<- error_event

Called for system errors

Class PanelClass event

Format `flag = this.error_event(error_object)`

Arguments error_object An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error

handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [PanelClass Methods](#) for more information.

-> initialize_event

Called before displaying a dialog box

Class PanelClass event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the panel dimensions, color, and so on.

See [PanelClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class PanelClass event

Format `this.resize_event(width, height, old_width, old_height)`

Arguments

<code>width</code>	The changed dialog box width.
<code>height</code>	The changed dialog box height.
<code>old_width</code>	The original dialog box width.
<code>old_height</code>	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [PanelClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class PanelClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [PanelClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class PanelClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [PanelClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class PanelClass event

Format this.update_event

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [PanelClass Methods](#) for more information.

<- arc_mode@

Returns arc mode setting

Class PenClass get

Format `val = this.arc_mode@`

Description The get method `arc_mode@` returns the arc mode setting. The setting indicates how filled arcs are drawn on the canvas with the CanvasClass method [fill_arc@](#). The valid arc mode values are:

0 XVAL_ArcChord, a chord is drawn between arc points.

1 XVAL_ArcPieSlice, a pie slice angle is drawn between the arc points.

You can use the method with the set method `arc_mode@` to verify and change the arc mode. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- background_color@

Returns background color setting

Class PenClass get

Format `backArray = this.background_color@`

Description The get method `background_color@` returns the background color setting as an array of values. The color settings are returned as follows:

`backArray[0]` The color type. The valid color types are:

0 RGB

2 CMYK

5 Named color

The following values apply to RGB color type:

`backArray[1]` The RGB red value.

backArray[2] The RGB blue value.

backArray[3] The RGB green value.

The following values apply to CMYK color type:

backArray[1] The CMYK cyan value.

backArray[2] The CMYK magenta value.

backArray[3] The CMYK yellow value.

backArray[4] The CMYK black value.

The following values apply to a named color type:

backArray[1] The color name.

You can use the method with the set method `background_color@` to verify and change the background color setting.

See [PenClass Methods](#) for more information.

<- cap_style@

Returns line endpoint style

Class PenClass get

Format value = this.cap_style@

Description The get method `cap_style@` returns the line endpoint style. The valid cap style values are:

- 0 XVAL_CapNotLast, the endpoint is not drawn for a line width of 0 or 1.
- 1 XVAL_CapButt, the endpoint is square with no projection beyond the endpoint.
- 2 XVAL_CapRound, the line is terminated with a circular arc with a diameter equal to the line width. The style acts as a XVAL_CapButt style for a line width of 0 or 1.
- 3 XVAL_CapProjecting, the line is terminated with a square extending beyond the endpoint. The square extends beyond the endpoint by half the line width. The style acts as a XVAL_CapButt style for a line width of 0 or 1.

You can use the method with the set method `cap_style@` to verify and change the cap style. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- clip_mask@

Returns clip mask

Class PenClass get

Format value = this.clip_mask@

Description The get method clip_mask@ returns the clip mask used for drawing in the canvas. The clip mask is the bitmap used to limit a drawing area. Pixels outside the bitmap area are not drawn. Pixels that are not set in the bitamp are not drawn.

You can use the method with the set method clip_mask@ to verify and change the clip mask.

See [PenClass Methods](#) for more information.

<- clip_x_origin@

Returns clip mask x-axis origin

Class PenClass get

Format value = this.clip_x_origin@

Description The get method clip_x_origin@ returns the clip mask x-axis origin. You can use the method with the set method clip_x_origin@ to verify and change the clip mask x-axis origin.

See [PenClass Methods](#) for more information.

<- clip_y_origin@

Returns clip mask y-axis origin

Class PenClass get

Format value = this.clip_y_origin@

Description The get method clip_y_origin@ returns the clip mask y-axis origin used for rectangles and arcs. You can use the method with the set method clip_y_origin to verify and change the clip mask y-axis origin.

See [PenClass Methods](#) for more information.

<- dashes@

Returns dash pattern

Class PenClass get

Format value = this.dashes@

Description The get method dashes@ returns the dash pattern as an integer. The value is the number of pixels in the dashes and gaps. You can use the method with the set method dashes@ to verify and change the dash pattern.

See [PenClass Methods](#) for more information.

<- dash_offset@

Returns dash pattern offset

Class PenClass get

Format value = this.dash_offset@

Description The get method dash_offset@ returns the dash pattern offset, in pixels, from the beginning of the line. You can use the method with the set method dash_offset@ to verify and change the dash offset.

See [PenClass Methods](#) for more information.

<- fill_rule@

Returns fill rule

Class PenClass get

Format value = this.fill_rule@

Description The get method fill_rule@ returns the fill rule used for polygons. The valid fill rule values are:

- 0 XVAL_EvenOddRule, areas that overlap an odd number of times are not filled.
- 1 XVAL_WindingRule, overlapping areas are always filled.

You can use the method with the set method `fill_rule@` to verify and change the fill rule. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- fill_style@

Returns fill style

Class PenClass get

Format value = this.fill_style@

Description The get method `fill_style@` returns the fill style used for rectangles and arcs. The valid fill style values are:

- 0 XVAL_FillSolid, fill graphic with foreground color.
- 1 XVAL_FillTiled, fill graphic object with the bitmap specified by the set method `tile`.
- 2 XVAL_FillStippled, fill graphic object with the 1 plane bitmap specified by the set method `stipple`.
- 3 XVAL_FillOpaqueStippled, fill graphic object with the 1 plane bitmap specified by the set method `stipple`. Use the foreground colors for set pixels in the bitmap. Use the background color for pixels that are not set in the bitmap.

You can use the method with the set method `fill_style@` to verify and change the fill rule. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- font_attrs@

Returns font attributes

Class PenClass get

Format format font_attrs_info@ info = this.font_attrs@

Description The get method `font_attrs@` returns the font attributes in `font_attrs_info@` format. The `font_attrs_info@` format is defined in the header file `builder_.am`, located in the `install_dir/axdata/elf` directory. The format is defined as:

format `font_attrs_info@`

<code>font_name,</code>	Font name
<code>font_size,</code>	Font size, in points.
<code>bold,</code>	A Boolean value, TRUE means the font is bold, otherwise FALSE.
<code>italic,</code>	A Boolean value, TRUE means the font is italic, otherwise FALSE.
<code>shadow</code>	Not used.

You can use the method with the set method `font_attrs@` to verify and change the font attributes. Use this method to retrieve all font attributes in one step. The values for font name and font size are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- font_bold@

Returns font bold state

Class PenClass get

Format flag = this.font_bold@

Description The get method `font_bold@` returns the font bold state as a Boolean value. TRUE means the font is bold, otherwise it returns FALSE. You can use the method with the set method `font_bold@` to verify and change the font bold state. Use the get method [font_attrs@](#) to retrieve all font information in one step.

See [PenClass Methods](#) for more information.

<- font_italic@

Returns font italic state

Class PenClass get

Format flag = this.font_italic@

Description The get method `font_italic@` returns the font italic state as a Boolean value. TRUE means the font is italic, otherwise it returns FALSE. You can use the method with the set method

`font_italic@` to verify and change the font italic state. Use the get method `font_attrs@` to retrieve all font information in one step.

See [PenClass Methods](#) for more information.

`<- font_name@`

Returns font

Class PenClass get

Format name = this.font_name@

Description The get method `font_name@` returns the font name as a string. The valid font names are:

default	XVAL_font_default, the default font.
Avant Garde	XVAL_font_avant_garde, the Avant Garde font.
Bookman	XVAL_font_bookman, the Bookman font.
Courier	XVAL_font_courier, the Courier font.
Helvetica	XVAL_font_helvetica, the Helvetica font.
Helvetica Narrow	XVAL_font_helvetica_narrow, the Helvetica Narrow font.
New Century Schoolbook	XVAL_font_new_century_schoolbook, the New Century Schoolbook font.
Palatino	XVAL_font_palatino, the Palatino font.
Symbol	XVAL_font_symbol, the Symbol font.
Times	XVAL_font_times, the Times font.
Zapf Chancery	XVAL_font_zapf_chancery, the Zapf Chancery font.
Zapf Dingbats	XVAL_font_zapf_dingbats, the Zapf Dingbats font.

You can use the method with the set method `font_name@` to verify and change the font. The values for font name are defined in the header file `elfpen_.am`. Use the get method `font_attrs@` to retrieve all font information in one step.

See [PenClass Methods](#) for more information.

<- font_shadow@

Returns font shadow state

Class PenClass get

Format flag = this.font_shadow@

Description The get method font_shadow@ returns the font shadow state as a Boolean value. TRUE means the font has a shadow, otherwise it returns FALSE. You can use the method with the set method font_shadow@ to verify and change the font shadow state. Use the get method [font_attrs@](#) to retrieve all font information in one step.

See [PenClass Methods](#) for more information.

<- font_size@

Returns font size

Class PenClass get

Format size = this.font_size@

Description The get method font_size@ returns the font size as a numeric value. The valid font sizes are:

10	XVAL_size_10, 10 point.
12	XVAL_size_12, 12 point.
14	XVAL_size_14, 14 point.
18	XVAL_size_18, 18 point.
24	XVAL_size_24, 24 point.
36	XVAL_size_36, 36 point.

You can use the method with the set method font_size@ to verify and change the font size. The values for font size are defined in the header file elfpen_.am. Use the get method [font_attrs@](#) to retrieve all font information in one step.

See [PenClass Methods](#) for more information.

<- foreground_color@

Returns foreground color setting

Class PenClass get

Format foreArray = this.foreground_color@

Description The get method foreground_color@ returns the foreground color setting as an array of values. The color settings are returned as follows:

foreArray[0] The color type. The valid color types are:

0 RGB

2 CMYK

The following values apply to RGB color type:

foreArray[1] The RGB red value.

foreArray[2] The RGB blue value.

foreArray[3] The RGB green value.

The following values apply to CMYK color type:

foreArray[1] The CMYK cyan value.

foreArray[2] The CMYK magenta value.

foreArray[3] The CMYK yellow value.

foreArray[4] The CMYK black value.

You can use the method with the set method foreground_color@ to verify and change the foreground color setting.

See [PenClass Methods](#) for more information.

<- graphics_exposures@

Returns graphics exposure state

Class PenClass get

Format flag = this.graphics_exposures@

Description The get method `graphics_exposures` returns the graphics exposure state as a Boolean value. TRUE means generate a graphics exposure event. FALSE means do not generate a graphics exposure event.

Two types of graphics exposure events can be generated.

Multiple graphics exposure events are generated when part of the canvas is obscured, unmapped, or unavailable,

A single graphics exposure event is generated when the entire canvas is available.

You can use the method with the set method `graphics_exposures@` to verify and change the graphics exposure state.

See [PenClass Methods](#) for more information.

<- join_style@

Returns join style

Class PenClass get

Format value = this.join_style@

Description The get method `join_style@` returns the join style used for drawing line corners. The valid join style values are:

- 0 XVAL_JoinMiter, the outer edges of lines extend and meet at an angle.
- 1 XVAL_JoinRound, the lines are joined by an arc with the line width diameter and centered on the join point.
- 2 XVAL_JoinBevel, the lines have a XVAL_CapButt cap style, with the area between outer edges filled.

You can use the method with the set method `join_style@` to verify and change the join style. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- line_style@

Returns line style

Class PenClass get

Format value = this.line_style@

Description The get method `line_style@` returns the line style. The valid line style values are:

- 0 XVAL_LineSolid, the line is continuously drawn with the foreground color.
- 1 XVAL_LineOnOffDash, the line dashes are drawn with the foreground color, the cap style is applied to each dash.
- 2 XVAL_LineDoubleDash, the line dashes are drawn with the foreground color, the line gaps are drawn with the background color, the XVAL_CapButt cap style is used where dashes and gaps meet.

You can use the method with the set method `line_style@` to verify and change the join style. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

`<- line_width@`

Returns line width

Class PenClass get

Format value = this.line_width@

Description The get method `line_width@` returns the line width, in pixels. You can use the method with the set method `line_width@` to verify and change the line width.

See [PenClass Methods](#) for more information.

`<- plane_mask@`

Returns plane mask value

Class PenClass get

Format value = this.plane_mask@

Description The get method `plane_mask@` returns the plane mask value as an unsigned long integer. The plane mask defines which planes of the display can be modified by a drawing command. All planes are modifiable, by default.

See [PenClass Methods](#) for more information.

<- stipple@

Returns stipple bitmap

Class PenClass get

Format bitmap = this.stipple@

Description The get method `stipple@` returns the name of the bitmap used to fill arcs, polygons, or rectangles. A stipple is a bitmap of depth 1, not the depth of the display.

See [PenClass Methods](#) for more information.

<- subwindow_mode@

Returns subwindow mode

Class PenClass get

Format mode = this.subwindow_mode@

Description The get method `subwindow_mode@` returns the subwindow mode. The subwindow mode controls the obscuring of parenting windows by subwindows. The valid subwindow modes are:

- 0 XVAL_ClipByChildren, drawing into an area obscured by a visible child has no effect. This is the default mode.
- 1 XVAL_IncludInferiors, drawing into an area obscured by a visible child appears through the child.

You can use the method with the set method `subwindow_mode@` to verify and change the subwindow mode. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- tile@

Returns tile bitmap

Class PenClass get

Format bitmap = this.tile@

Description The get method `tile@` returns the name of the bitmap used to fill arcs, polygons, or rectangles. A tile is a bitmap of depth 1 or more. A tile of depth 1 and drawing plane of depth 1 acts the same as a stipple.

See [PenClass Methods](#) for more information.

<- `ts_x_origin@`

Returns tile x-axis origin

Class PenClass get

Format value = `this.ts_x_origin@`

Description The get method `ts_x_origin@` returns the x-axis origin of a tile relative to the origin of the arc, polygon, or rectangle.

See [PenClass Methods](#) for more information.

<- `ts_y_origin@`

Returns tile y-axis origin

Class PenClass get

Format value = `this.ts_y_origin@`

Description The get method `ts_y_origin@` returns the y-axis origin of a tile relative to the origin of the arc, polygon, or rectangle.

See [PenClass Methods](#) for more information.

<- `xfer_func@`

Returns logical function value

Class PenClass get

Format value = `this.xfer_func@`

Description The get method `xfer_func@` returns the logical function value. The logical function controls how source drawing pixels are combined with existing destination pixel values.

Logical functions are also known as raster operations, raster functions, or display functions. The valid logical function values are:

- 0 XVAL_GXclear, 0.
- 1 XVAL_GXand, source AND destination.
- 2 XVAL_GXandReverse, source AND (NOT destination).
- 3 XVAL_GXcopy, source.
- 4 XVAL_GXandInverted, (NOT source) AND destination.
- 5 XVAL_GXnoop, destination.
- 6 XVAL_GXxor, source XOR destination.
- 7 XVAL_GXor, source OR destination.
- 8 XVAL_GXnor, (NOT source) AND (NOT destination).
- 9 XVAL_GXequiv, (NOT source) XOR destination.
- 10 XVAL_GXinvert, NOT destination.
- 11 XVAL_GXorReverse, source OR (NOT destination).
- 12 XVAL_GXcopyInverted, (NOT source).
- 13 XVAL_GXorInverted, (NOT source) OR destination.
- 14 XVAL_GXnand, (NOT source) OR (NOT destination).
- 15 XVAL_GXset, 1.

You can use the method with the set method `xfer_func@` to verify and change the `xfer_func@` function. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> arc_mode@

Sets arc mode

Class PenClass set

Format this.arc_mode@(value)

Arguments value The arc mode setting. The valid arc mode settings are:
XVAL_ArcChord A chord is drawn between arc points.
XVAL_ArcPieSlice A pie slice angle is drawn between the arc points.

Description The set method `arc_mode@` sets the arc mode setting. The setting indicates how filled arcs are drawn on the canvas with the CanvasClass method [fill_arc@](#).

You can use the method with the get method `arc_mode@` to verify and change the arc mode. The settings are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> background_color@

Sets background color

Class PenClass set

Format `this.background_color@(backArray)`

Arguments backArray An array of color settings. The array contains the following information:

backArray[0] The color type. The valid color types are:

- 0 RGB
- 2 CMYK
- 5 Named color

The following values apply to RGB color type:

backArray[1] The RGB red value.

backArray[2] The RGB blue value.

backArray[3] The RGB green value.

The following values apply to CMYK color type:

backArray[1] The CMYK cyan value.

backArray[2] The CMYK magenta value.

backArray[3] The CMYK yellow value.

backArray[4] The CMYK black value.

The following values apply to a named color type:

backArray[1] The color name.

Description The set method `background_color@` sets the background color with an array of values. You can use the method with the get method `background_color@` to verify and change the background color setting.

See [PenClass Methods](#) for more information.

-> `background_color_cmyk@`

Sets background CMYK color

Class PenClass set

Format `this.background_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `background_cmyk@` sets the background color with CMYK values. You can use the method with the get method `background_cmyk@` to verify and change the background color setting.

See [PenClass Methods](#) for more information.

-> `background_color_name@`

Sets background color by name

Class PenClass set

Format `this.background_color_name@(name)`

Arguments `name` The color name.

Description The set method `background_name@` sets the background color by color name. You can use the method with the get method `background_name@` to verify and change the background color setting.

See [PenClass Methods](#) for more information.

-> background_color_rgb@

Sets background RGB color

Class PenClass set

Format this.background_color_rgb@(red, green, blue)

Arguments

red	The RGB red value.
green	The RGB blue value.
blue	The RGB green value.

Description The set method background_rgb@ sets the background color with RGB values. You can use the method with the get method background_rgb@ to verify and change the background color setting.

See [PenClass Methods](#) for more information.

-> cap_style@

Sets line endpoint style

Class PenClass set

Format this.cap_style@(value)

Arguments

value	The line endpoint setting. The valid endpoint settings are:	
	XVAL_CapNotLast	The endpoint is not drawn for a line width of 0 or 1.
	XVAL_CapButt	The endpoint is square with no projection beyond the endpoint.
	XVAL_CapRound	The line is terminated with a circular arc with a diameter equal to the line width. The style acts as a XVAL_CapButt style for a line width of 0 or 1.
	XVAL_CapProjecting	The line is terminated with a square extending beyond the endpoint. The square extends beyond the endpoint by half the line width. The style acts as a XVAL_CapButt style for a line width of 0 or 1.

Description The set method cap_style@ sets the line endpoint style. You can use the method with the get method cap_style @to verify and change the cap style. The values are defined in the header file elfpen_.am.

See [PenClass Methods](#) for more information.

-> clip_mask@

Sets clip mask

Class PenClass set

Format this.clip_mask@(bitmap)

Arguments bitmap A bitmap located in your ELF search path, or loaded as a resource.

Description The set method clip_mask@ sets the clip mask used for drawing in the canvas. The clip mask is the bitmap used to limit a drawing area. Pixels outside the bitmap area are not drawn. Pixels that are not set in the bitamp are not drawn.

You can use the method with the get method clip_mask@ to verify and change the clip mask.

See [PenClass Methods](#) for more information.

-> clip_x_origin@

Sets clip mask x-axis origin

Class PenClass set

Format this.clip_x_origin@(value)

Arguments value The x-axis position relative to the upper left corner of the canvas.

Description The set method clip_x_origin@ sets the clip mask x-axis origin. You can use the method with the get method clip_x_origin@ to verify and change the clip mask x-axis origin.

See [PenClass Methods](#) for more information.

-> clip_y_origin@

Sets clip mask y-axis origin

Class PenClass set

Format this.clip_y_origin@(value)

Arguments value The y-axis position relative to the upper left corner of the canvas.

Description The set method clip_y_origin@ sets the clip mask y--axis origin used for rectangles and arcs. You can use the method with the get method clip_y_origin@ to verify and change the clip mask y-axis origin.

See [PenClass Methods](#) for more information.

-> dashes@

Sets dash pattern

Class PenClass set

Format this.dashes@(value)

Arguments value The length, in pixels, of the dashes and gaps of the line pattern.

Description The set method dashes@ sets the line dash pattern. You can use the method with the get method dashes@ to verify and change the dash pattern.

See [PenClass Methods](#) for more information.

-> dash_offset@

Sets dash offset

Class PenClass set

Format this.dash_offset@(value)

Arguments value The offset, in pixels, of the dash pattern from the beginning of the line.

Description The set method dash_offset@ sets the offset of the dash pattern from the beginning of the line. You can use the method with the get method dash_offset@ to verify and change the dash offset.

See [PenClass Methods](#) for more information.

-> fill_rule@

Sets fill rule

Class PenClass set

Format this.fill_rule@(value)

Arguments value The fill rule setting. The valid fill rule settings are:

XVAL_EvenOddRule	Areas that overlap an odd number of times are not filled.
XVAL_WindingRule	Overlapping areas are always filled.

Description The set method fill_rule@ sets the fill rule used for polygons. You can use the method with the get method fill_rule@ to verify and change the fill rule. The values are defined in the header file elfpen_.am.

See [PenClass Methods](#) for more information.

-> fill_style@

Sets fill style

Class PenClass set

Format this.fill_style@(value)

Arguments value The fill style value. The valid fill style values are:

XVAL_FillSolid	Fill graphic with foreground color.
XVAL_FillTiled	Fill graphic object with the bitmap specified by the get method tile.
XVAL_FillStippled	Fill graphic object with the 1 plane bitmap specified by the get method stipple.
XVAL_FillOpaqueStippled	Fill graphic object with the 1 plane bitmap specified by the get method stipple. Use the foreground colors for get pixels in the bitmap. Use the background color for pixels that are not get in the bitmap.

Description The set method `fill_style@` sets the fill style used for rectangles and arcs. You can use the method with the get method `fill_style@` to verify and change the fill rule. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> font_attrs@

Sets font attributes

Class PenClass set

Format `this.font_attrs@(format font_attrs_info@ info)`

Arguments info The font attributes. The `font_attrs_info@` format is defined in the header file `builder_.am`, located in the `install_dir/axdata/elf` directory. The format is defined as:

format `font_attrs_info@`

font_name, Font name

font_size, Font size, in points.

bold, A Boolean value, TRUE means the font is bold, otherwise FALSE.

italic, A Boolean value, TRUE means the font is italic, otherwise FALSE.

shadow Not used.

Description The set method `font_attrs@` sets the font attributes. You can use the method with the get method `font_attrs@` to verify and change the font attributes. The values for font name and font size are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> font_bold@

Sets font bold state

Class PenClass set

Format `this.font_bold@(flag)`

Arguments flag A Boolean value. TRUE means the font is bold, otherwise it sets FALSE.

Description The set method `font_bold@` sets the font bold state. You can use the method with the get method `font_bold@` to verify and change the font bold state.

See [PenClass Methods](#) for more information.

-> font_italic@

Sets font italic state

Class PenClass set

Format `this.font_italic@(flag)`

Arguments `flag` A Boolean value. TRUE means the font is italic, otherwise it sets FALSE.

Description The set method `font_italic@` sets the font italic state. You can use the method with the get method `font_italic@` to verify and change the font italic state.

See [PenClass Methods](#) for more information.

-> font_name@

Sets font

Class PenClass set

Format `this.font_name@(name)`

Arguments `name` The font name. The valid font names are:

<code>XVAL_font_default</code>	Default font.
<code>XVAL_font_avant_garde</code>	Avant Garde font.
<code>XVAL_font_bookman</code>	Bookman font.
<code>XVAL_font_courier</code>	Courier font.
<code>XVAL_font_helvetica</code>	Helvetica font.
<code>XVAL_font_helvetica_narrow</code>	Helvetica Narrow font.
<code>XVAL_font_new_century_schoolbook</code>	New Century Schoolbook font.
<code>XVAL_font_palatino</code>	Palatino font.
<code>XVAL_font_symbol</code>	Symbol font.
<code>XVAL_font_times</code>	Times font.

XVAL_font_zapf_chancery	Zapf Chancery font.
XVAL_font_zapf_dingbats	Zapf Dingbats font.

Description The set method `font_name@` sets the font name as a string. You can use the method with the get method `font_name@` to verify and change the font. The values for font name are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> font_shadow@

Sets font shadow state

Class PenClass set

Format `this.font_shadow@(flag)`

Arguments `flag` A Boolean value. TRUE means the font has a shadow, otherwise it sets FALSE.

Description The set method `font_shadow@` sets the font shadow state. You can use the method with the get method `font_shadow@` to verify and change the font shadow state.

See [PenClass Methods](#) for more information.

-> font_size@

Sets font size

Class PenClass set

Format `this.font_size@(size)`

Arguments `size` The font size. The defined font sizes are as follows, but you can use any positive value:

XVAL_size_10	10 point.
XVAL_size_12	12 point.
XVAL_size_14	14 point.
XVAL_size_18	18 point.
XVAL_size_24	24 point.
XVAL_size_36	36 point.

Description The set method `font_size@` sets the font size. You can use the method with the get method `font_size@` to verify and change the font size. The values for font size are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> foreground_color@

Sets foreground color

Class PenClass set

Format `this.foreground_color@(backArray)`

Arguments `backArray` An array of color settings. The array contains the following information:

`backArray[0]` The color type. The valid color types are:

- 0 RGB
- 2 CMYK
- 5 Named color

The following values apply to RGB color type:

`backArray[1]` The RGB red value.

`backArray[2]` The RGB blue value.

`backArray[3]` The RGB green value.

The following values apply to CMYK color type:

`backArray[1]` The CMYK cyan value.

`backArray[2]` The CMYK magenta value.

`backArray[3]` The CMYK yellow value.

`backArray[4]` The CMYK black value.

The following values apply to a named color type:

`backArray[1]` The color name.

Description The set method `foreground_color@` sets the foreground color with an array of values. You can use the method with the get method `foreground_color@` to verify and change the foreground color setting.

See [PenClass Methods](#) for more information.

-> foreground_color_cmyk@

Sets foreground CMYK color

Class PenClass set

Format this.foreground_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method foreground_cmyk@ sets the foreground color with CMYK values. You can use the method with the get method foreground_cmyk@ to verify and change the foreground color setting.

See [PenClass Methods](#) for more information.

-> foreground_color_name@

Sets foreground color by name

Class PenClass set

Format this.foreground_color_name@(name)

Arguments name The color name.

Description The set method foreground_name@ sets the foreground color by color name. You can use the method with the get method foreground_name@ to verify and change the foreground color setting.

See [PenClass Methods](#) for more information.

-> foreground_color_rgb@

Sets foreground RGB color

Class PenClass set

Format this.foreground_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method foreground_rgb@ sets the foreground color with RGB values. You can use the method with the get method foreground_rgb@ to verify and change the foreground color setting.

See [PenClass Methods](#) for more information.

-> graphics_exposures@

Sets graphics exposure state

Class PenClass set

Format this.graphics_exposures@(flag)

Arguments flag A Boolean value. TRUE means generate a graphics exposure event.
 FALSE means do not generate a graphics exposure event.

Description The set method graphics_exposures@ sets the graphics exposure state. Two types of graphics exposure events can be generated.

Multiple graphics exposure events are generated when part of the canvas is obscured, unmapped, or unavailable,

A single graphics exposure event is generated when the entire canvas is available.

You can use the method with the get method graphics_exposures@ to verify and change the graphics exposure state.

See [PenClass Methods](#) for more information.

-> join_style@

Sets join style

Class PenClass set

Format this.join_style@(value)

Arguments value The join style. The valid join style values are:

XVAL_JoinMiter	The outer edges of lines extend and meet at an angle.
XVAL_JoinRound	The lines are joined by an arc with the line width diameter and centered on the join point.
XVAL_JoinBevel	The lines have a XVAL_CapButt cap style, with the area between outer edges filled.

Description The set method `join_style@` sets the join style used for drawing line corners. You can use the method with the get method `join_style` to verify and change the join style. The values are defined in the header file `elfpen_.am`

See [PenClass Methods](#) for more information.

-> line_style@

Sets line style

Class PenClass set

Format `this.line_style@(value)`

Arguments value The line style. The valid line style values are:

XVAL_LineSolid	The line is continuously drawn with the foreground color.
XVAL_LineOnOffDash	The line dashes are drawn with the foreground color, the cap style is applied to each dash.
XVAL_LineDoubleDash	The line dashes are drawn with the foreground color, the line gaps are drawn with the background color, the XVAL_CapButt cap style is used where dashes and gaps meet.

Description The set method `line_style@` sets the line style. You can use the method with the get method `line_style@` to verify and change the join style. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> line_width@

Sets line width

Class PenClass set

Format this.line_width@(value)

Arguments value The line width, in pixels.

Description The set method line_width@ sets the line width, in pixels. You can use the method with the get method line_width@ to verify and change the line width.

See [PenClass Methods](#) for more information.

-> plane_mask@

Sets plane mask value

Class PenClass set

Format this.plane_mask@(value)

Arguments value An unsigned long integer. By default all planes are available. A value of 0 allows no changes to any plane. A value of 1 allows changes to a single plane. A value of -1 restores the default of all planes.

Description The set method plane_mask@ sets the plane mask value. The plane mask defines which planes of the display can be modified by a drawing command.

See [PenClass Methods](#) for more information.

-> stipple@

Sets stipple bitmap

Class PenClass set

Format this.stipple@(bitmap)

Arguments bitmap A bitmap located in your ELF search path, or loaded as a resource.

Description The set method `stipple@` sets the name of the bitmap used to fill arcs, polygons, or rectangles. A stipple is a bitmap of depth 1, not the depth of the display.

See [PenClass Methods](#) for more information.

-> `subwindow_mode@`

Sets subwindow mode

Class PenClass set

Format `this.subwindow_mode@(mode)`

Arguments mode The subwindow mode. The valid modes are:

XVAL_ClipByChildren	Drawing into an area obscured by a visible child has no effect. This is the default mode
XVAL_IncludeInferiors	Drawing into an area obscured by a visible child appears through the child.

Description The set method `subwindow_mode@` sets the subwindow mode. The subwindow mode controls the obscuring of parenting windows by subwindows. You can use the method with the get method `subwindow_mode@` to verify and change the subwindow mode. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

-> `tile@`

Sets tile bitmap

Class PenClass set

Format `this.tile@(bitmap)`

Arguments bitmap A bitmap located in your ELF search path, or loaded as a resource.

Description The set method `tile@` sets the name of the bitmap used to fill arcs, polygons, or rectangles. A tile is a bitmap of depth 1 or more. A tile of depth 1 and drawing plane of depth 1 acts the same as a stipple.

See [PenClass Methods](#) for more information.

-> **ts_x_origin@**

Sets tile x-axis origin

Class PenClass set

Format this.ts_x_origin@(value)

Arguments value The x-axis origin of a tile relative to the origin of the arc, polygon, or rectangle.

Description The set method ts_x_origin@ sets the x-axis origin of a tile relative to the origin of the arcs, polygons, or rectangles.

See [PenClass Methods](#) for more information.

-> **ts_y_origin@**

Sets tile y-axis origin

Class PenClass set

Format this.ts_y_origin@(value)

Arguments value The y-axis origin of a tile relative to the origin of the arc, polygon, or rectangle.

Description The set method ts_y_origin@ sets the y-axis origin of a tile relative to the origin of the arcs, polygons, or rectangles.

See [PenClass Methods](#) for more information.

-> **xfer_func@**

Sets logical function value

Class PenClass set

Format this.xfer_func@(value)

Arguments value The logical function. The valid logical functions are:

9. XVAL_GXclear

XVAL_GXand	source AND destination.
XVAL_GXandReverse	source AND (NOT destination).
XVAL_GXcopy	source.
XVAL_GXandInverted	(NOT source) AND destination.
XVAL_GXnoop	destination.
XVAL_GXxor	source XOR destination.
XVAL_GXor	source OR destination.
XVAL_GXnor	(NOT source) AND (NOT destination).
XVAL_GXequiv	(NOT source) XOR destination.
XVAL_GXinvert	NOT destination.
XVAL_GXorReverse	source OR (NOT destination).
XVAL_GXcopyInverted	(NOT source).
XVAL_GXorInverted	(NOT source) OR destination.
XVAL_GXnand	(NOT source) OR (NOT destination).

1. XVAL_GXset

Description The set method `xfer_func@` sets the logical function value. The logical function controls how source drawing pixels are combined with existing destination pixel values. Logical functions are also known as raster operations, raster functions, or display functions. You can use the method with the get method `xfer_func@` to verify and change the `xfer_func@` function. The values are defined in the header file `elfpen_.am`.

See [PenClass Methods](#) for more information.

<- error_event

Called for system errors

Class PenClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [PenClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class PenClass event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [PenClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class PenClass event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition.

See [PenClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class PenClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [PenClass Methods](#) for more information.

<- copies@

Returns number of copies

Class PrinterClass get

Format qty = this.copies@

Description The get method copies@ returns the number of print job copies as an integer.

See [PrinterClass Methods](#) for more information.

<- cursor@

Returns cursor position on print area

Class PrinterClass get

Format pointArray = this.cursor@

Description The get method `cursor@` returns the cursor position on the print area as an array. The cursor is used as the point of origin for the drawing methods. The array contains the following values:

<code>pointArray[0]</code>	The x-axis coordinate relative to the left margin, in mils. (1000 mils = 1 inch)
<code>pointArray[1]</code>	The y-axis coordinate relative to the top margin, in mils.

See [PrinterClass Methods](#) for more information.

<- end_page@

Returns end page number

Class PrinterClass get

Format `pageNum = this.end_page@`

Description The get method `end_page@` returns the end page number. Use the `end_page@` and `start_page@` methods to determine the print job page range. If `end_page@` is an integer less than `start_page@`, then the page range is all pages.

See [PrinterClass Methods](#) for more information.

<- icon_size@

Returns icon size

Class PrinterClass get

Format `sizeArray = this.icon_size@(object pen, name)`

Arguments

<code>pen</code>	A pen object.
<code>name</code>	The icon name, less the file extension. The icon is a .im file located in your ELF search path, or a glom file loaded as a resource.

Description The get method `icon_size@` returns the size of an icon in the print area drawn with a given pen object. The size is returned as a two element array. `sizeArray[0]` is the icon width, in mils (1000 mils = 1 inch), and `sizeArray[1]` is the icon height, in mils. The size of the icon in the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

<- is_print_banner@

Returns banner page status

Class PrinterClass get

Format flag = this.is_print_banner@

Description The get method is_print_banner@ returns the banner page status. TRUE means banner pages are enabled. FALSE means banner pages are disabled. Use banner pages to print a special page at the beginning of your document to help separate your document from other printer jobs.

See [PrinterClass Methods](#) for more information.

<- is_print_collated@

Returns collating status

Class PrinterClass get

Format flag = this.is_print_collated@

Description The get method is_print_collated@ returns the banner page status. TRUE means collating is enabled. FALSE means collating is disabled. Use collating to put multiple copies in order as sets, such as page 1,2,3,1,2,3 and so on. When collating is set, printing performance will be slower.

See [PrinterClass Methods](#) for more information.

<- is_print_in_background@

Returns background printing status

Class PrinterClass get

Format flag = this.is_print_in_background@

Description The get method is_print_in_background@ returns the background printing status. TRUE means background printing is enabled. FALSE means background printing is

disabled. You can print information in the background, instead of waiting for the print job to finish before continuing application processing.

See [PrinterClass Methods](#) for more information.

<- is_print_in_color@

Returns print color status

Class PrinterClass get

Format flag = this.is_print_in_color@

Description The get method is_print_in_color@ returns the color status. TRUE means print the file on a color printer. FALSE means print on a non-color printer. If the option is set to TRUE, but the printer is a non-color printer, the colors are translated into gray scale values.

See [PrinterClass Methods](#) for more information.

<- is_started@

Returns printing status

Class PrinterClass get

Format startFlag = this.is_started@

Description The get method is_started@ returns the printing status as a Boolean value. The method returns TRUE if printing is started, otherwise the method returns FALSE.

See [PrinterClass Methods](#) for more information.

<- page_dimensions@

Returns print area dimensions

Class PrinterClass get

Format pageArray = this.page_dimensions@

Description The get method page_dimensions@ returns the print area dimensions as an array. The array contains the following values:

pageArray[0] The print area width, in mils. (1000 mils = 1 inch)
pageArray[1] The print area height, in mils.

See [PrinterClass Methods](#) for more information.

<- page_height@

Returns print area height

Class PrinterClass get

Format height = this.page_height@

Description The get method page_height@ returns the print area height, in mils. (1000 mils = 1 inch)

See [PrinterClass Methods](#) for more information.

<- page_margins@

Returns print area margins

Class PrinterClass get

Format marginArray = this.page_margins@

Description The get method page_margins@ returns the print area margins as an array. The array contains the following values:

marginArray[0] The left margin, in mils. (1000 mils = 1 inch)
marginArray[1] The right margin, in mils.
marginArray[2] The top margin, in mils.
marginArray[3] The bottom margin, in mils.

See [PrinterClass Methods](#) for more information.

<- page_number@

Returns current print page

Class PrinterClass get

Format page = this.page_number@

Description The get method page_number@ returns the current print page. Page numbering starts at 1. Use the set method new_page@ to end the current page and start a new page.

See [PrinterClass Methods](#) for more information.

<- page_orientation@

Returns page orientation

Class PrinterClass get

Format orient = this.page_orientation@

Description The get method page_orientation@ returns the print page orientation as a string. Page orientation applies to pages defined with a standard page size. Page orientation is either Portrait, printed across the shortest dimension of the page, or Landscape, printed across the longest dimension of the page.

See [PrinterClass Methods](#) for more information.

<- page_size@

Returns print page size

Class PrinterClass get

Format size = this.page_size@

Description The get method page_size@ returns the print page size. Page sizes are defined as follows:

Letter	8500 mils W x 1100 mils H. (1000 mils = 1 inch)
Legal	8500 mils W x 1400 mils H.
Ledger	1700 mils W x 1100 mils H.
Executive	7500 mils W x 1000 mils H.
A3	11694 mils W x 16528 mils H.
A4	8264 mils W x 11694 mils H.
Custom	Page is set to a custom width and height with the set method page_dimensions@.

See [PrinterClass Methods](#) for more information.

<- page_width@

Returns print area width

Class PrinterClass get

Format width = this.page_width@

Description The get method page_width@ returns the print area width, in mils. (1000 mils = 1 inch)

See [PrinterClass Methods](#) for more information.

<- preview_page_number@

Returns print preview page

Class PrinterClass get

Format page = this.preview_page_number@

Description The get method preview_page_number@ returns the print page displayed in the print preview dialog box. Page numbering starts at 1.

See [PrinterClass Methods](#) for more information.

<- printer_name@

Returns destination printer

Class PrinterClass get

Format printer = this.printer_name@

Description The get method printer_name@ returns the destination printer of the print job. If the printer is NULL, then the print job is printed to file.

See [PrinterClass Methods](#) for more information.

<- printer_type@

Returns printer class

Class PrinterClass get

Format class = this.printer_type@

Description The get method printer_type@ returns the printer type as a string. The valid printer types are either PostScript or PCL5.

See [PrinterClass Methods](#) for more information.

<- print_file_name@

Returns print file name

Class PrinterClass get

Format file = this.print_file_name@

Description The get method print_file_name@ returns the print file name. If printer_name@ is set to NULL, the print job is printed to file, otherwise the file name is a temporary file spooled to the printer.

See [PrinterClass Methods](#) for more information.

<- start_page@

Returns start page number

Class PrinterClass get

Format class = this.start_page@

Description The get method start_page@ returns the start page number. Use the end_page@ and start_page@ methods to determine the print job page range. If end_page@ is an integer less than start_page@, then the page range is all pages.

See [PrinterClass Methods](#) for more information.

<- text_size@

Returns text size

Class PrinterClass get

Format sizeArray = this.text_size@(object pen, text)

Arguments pen The PenClass object used to write the text.
text A text string, array of text strings, or 2-dimensional array of text strings.

Description The get method text_size@ returns the potential size of text drawn in a print area with a given PenClass object. The method returns a an array with the following values:
sizeArray[0] The text width, in mils. (1000 mils = 1 inch)
sizeArray[1] The text height, in mils.
sizeArray[2] The ascender height, in mils.

See [PrinterClass Methods](#) for more information.

-> copies@

Sets number of print copies

Class PrinterClass set

Format this.copies@(qty)

Arguments qty The number of print job copies. The number of copies must be an integer greater than or equal to 1. The default number of copies is 1.

Description The set method copies@ sets the number of print copies.

See [PrinterClass Methods](#) for more information.

-> cursor@

Sets cursor position on print area

Class PrinterClass set

Format this.cursor@(pointArray)

Arguments pointArray A two-element array containing the following values:

pointArray[0]	The x-axis coordinate relative to the left margin, in mils. (1000 mils = 1 inch)
pointArray[1]	The y-axis coordinate relative to the top margin, in mils.

Description The set method cursor@ sets the cursor position on the print area. The cursor is used as the point of origin for the drawing methods.

See [PrinterClass Methods](#) for more information.

-> defaults@

Sets printing default values

Class PrinterClass set

Format this.defaults@

Description The set method defaults@ sets the printing default values. The default values are:

started	FALSE
printer type	PostScript
preview	TRUE
page size	Letter
page height	11000 mils (1000 mils = 1 inch)
page width	8500 mils
left margin	500 mils
right margin	500 mils
top margin	1000 mils
bottom margin	1000 mils
header	NULL
footer	NULL
page number	0

See [PrinterClass Methods](#) for more information.

-> draw_arc@

Draws an arc

Class PrinterClass set

Format this.draw_arc@(object pen, x, y, width, height, startAngle, endAngle)

Arguments

pen	A pen object.
x	The upper left corner x position, in mils (1000 mils = 1 inch), of the rectangle containing the arc.
y	The upper left corner y position, in mils, of the rectangle containing the arc.
width	The width, in mils, of the rectangle containing the arc. The width is also known as the major axis of the arc.
height	The height, in mils, of the rectangle containing the arc. The height is also known as the minor axis of the arc.
startAngle	The starting angle, in quad degrees, of the arc, relative to the 3'o clock position from center.
endAngle	The ending angle, in quad degrees, of the arc, relative to the 3'o clock position from center. 1440 quad degrees is a complete circle.

Description The set method draw_arc@ draws an arc on the print area. The appearance of the arc on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> draw_chart@

Draws a chart

Class PrinterClass set

Format this.draw_chart@(object chart, width, height)

Arguments

chart	A ChartClass object in the application.
width	The width, in mils (1000 mils = 1 inch), of the chart in the printable area.
height	The height, in mils, of the chart in the printable area.

Description The set method `draw_chart@` draws a chart on the print area at the current cursor location.

See [PrinterClass Methods](#) for more information.

-> draw_icon@

Draws an icon

Class PrinterClass set

Format `this.draw_icon@(object pen, name)`

Arguments

pen	A pen object.
name	The icon name, less the file extension. The icon is a .im Applixware bit-map file or an image in a glom file. The files are located in your ELF search path, or are loaded as a resource in the Builder file.

Description The set method `draw_icon@` draws an icon on the print area at the current cursor location.

See [PrinterClass Methods](#) for more information.

-> draw_line@

Draws a line

Class PrinterClass set

Format `this.draw_line@(object pen, endPoint)`

Arguments

pen	A pen object.
endPoint	A two element array for the end point of the line. <code>point[0]</code> is the x position, in mils (1000 mils = 1 inch). <code>point[1]</code> is the y position, in mils.

Description The set method `draw_line@` draws a line on the print area, from the cursor to the end-point. The appearance of the line on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> draw_polygon@

Draws a polygon

Class PrinterClass set

Format this.draw_polygon@(object pen, points, quantity)

Arguments

pen	A pen object.
points	An array of points. Each point is an array of two elements. points[z,0] is the x position, in mils (1000 mils = 1 inch). points[z,1] is the y position, in mils.
quantity	The number of points in the points array.

Description The set method draw_polygon@ draws a polygon on the print area. The appearance of the polygon on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> draw_rectangle@

Draws a rectangle

Class PrinterClass set

Format this.draw_rectangle@(object pen, width, height)

Arguments

pen	A pen object.
width	The width, in mils, of the rectangle. (1000 mils = 1 inch)
height	The height, in mils, of the rectangle.

Description The set method draw_rectangle@ draws a rectangle on the print area. The upper left corner of the rectangle is at the current cursor position. The appearance of the rectangle on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> draw_text@

Draws text

Class PrinterClass set

Format this.draw_text@(object pen, text, advance)

Arguments

pen	A pen object.
text	A text string, an array of strings, or a 2-dimensional array of strings.
advance	An integer that sets the auto-advance characteristics of the cursor.

Description The set method draw_text@ draws text on the print area. The appearance of the text on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

If a single text string is passed, the text string is drawn at the cursor.

If an array of strings is passed, the first array item is drawn at the cursor. Succeeding array items are drawn on separate lines, advancing each line by the text height. Use the text_size@ method to get the text height. If the cursor advances into the bottom margin area, the new_page@ method is invoked, and subsequent lines of text are drawn on the new page. The cursor is left at the starting position of the last text string.

If a 2-dimensional array of strings is passed, the text is printed in a table format, with one row for each element of the first array dimension. Starting at the cursor, the area between the cursor and the right margin is divided into n equal sized areas, where n is the numbers of elements in the second dimension. Each text string is left justified in its area. Each row advances by the text height.

The advance argument determines the cursor position after the text is drawn. If advance is 0 or NULL, the cursor position is unchanged.

If advance is 1, the cursor x-position is unchanged, while the cursor y-position is advanced to the next line.

If advance is 2, the cursor x-position is set to the end of the line, while the cursor y-position is unchanged. If the text is an array of strings, the cursor x-position is set to the end of the longest line.

See [PrinterClass Methods](#) for more information.

-> end_page@

Sets end page number

Class PrinterClass set

Format this.end_page@(num)

Arguments num The end page number. The default end page is 0.

Description The set method end_page@ sets the end page number. The end_page@ and start_page@ methods determine the print job page range. If end_page@ is set to an integer less than start_page@, then the page range is all pages.

See [PrinterClass Methods](#) for more information.

-> fill_arc@

Draws a filled arc

Class PrinterClass set

Format this.fill_arc@(object pen, width, height, startAngle, endAngle)

Arguments

pen	A pen object.
width	The width, in mils (1000 mils = 1 inch), of the rectangle containing the arc. The width is also known as the major axis of the arc.
height	The height, in mils, of the rectangle containing the arc. The height is also known as the minor axis of the arc.
startAngle	The starting angle, in quad degrees, of the arc, relative to the 3'o clock position from center.
endAngle	The ending angle, in quad degrees, of the arc, relative to the 3'o clock position from center. 1440 quad degrees is a complete circle.

Description The set method fill_arc@ draws a filled arc on the print area at the current cursor position. The appearance of the filled arc on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> fill_polygon@

Draws a filled polygon

Class PrinterClass set

Format this.fill_polygon@(object pen, points, quantity)

Arguments

pen	A pen object.
points	An array of points. Each point is an array of two elements. points[z,0] is the x position, in mils (1000 mils = 1 inch). points[z,1] is the y position, in mils.
quantity	The number of points in the points array.

Description The set method fill_polygon@ draws a filled polygon on the print area. The appearance of the filled polygon on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> fill_rectangle@

Draws a filled rectangle

Class PrinterClass set

Format this.fill_rectangle@(object pen, width, height)

Arguments

pen	A pen object.
width	The width, in mils (1000 mils = 1 inch), of the rectangle.
height	The height, in mils, of the rectangle.

Description The set method fill_rectangle@ draws a filled rectangle on the print area at the current cursor position. The appearance of the filled rectangle on the print area is dependent on the pen object attributes. See [PenClass Methods](#) for more information on setting pen object attributes.

See [PrinterClass Methods](#) for more information.

-> footer@

Sets page footer

Class PrinterClass set

Format this.footer@(object lpen, ltext, object cpen, ctext, object rpen, rtext, point)

Arguments

lpen	A pen object.
ltext	The text string appearing in the footer, left aligned at the left margin.
cpen	A pen object.
ctext	The text string appearing in the footer, center aligned between the margins.
rpen	A pen object.
rtext	The text string appearing in the footer, right aligned at the right margin.
point	A two element array for the footer position, from the lower left corner of the page. point[0], the x position, is not used. point[1] is the y position, in mils (1000 mils = 1 inch), from the bottom margin.

Description The set method footer@ sets the footer text and position. Use the footer@ and header@ methods to define the footer and header on the first page before using the start_job@ method. Use the footer@ and header@ methods to define the footer and header on a subsequent page before using the new_page@ method.

Footer text has special formatting sequences. The sequence #p places the current page number in the footer text. The sequence #P places the last page number in the footer text. The sequence #dnnn places the date and time in the footer text. The date and time used is the time at which the [start_job@](#) method is invoked. The string nnn is the format for the date, as defined in the datetim_.am file, located in the */install_dir/axdata/elf* directory. You do not need to include the datetim_.am file in your object method source to place a date and time in the footer text.

See [PrinterClass Methods](#) for more information.

-> header@

Sets page header

Class PrinterClass set

Format this.header@(object lpen, ltext, object cpen, ctext, object rpen, rtext, point)

Arguments

lpen	A pen object.
ltext	The text string appearing in the footer, left aligned at the left margin.
cpen	A pen object.
ctext	The text string appearing in the footer, center aligned between the margins.
rpen	A pen object.
rtext	The text string appearing in the footer, right aligned at the right margin.
point	A two element array for the header position, from the upper left corner of the page. point[0], the x position, is not used. point[1] is the y position, in mils (1000 mils = 1 inch), from the top of the page.

Description The set method header@ sets the header text and position. Use the footer@ and header@ methods to define the footer and header on the first page before using the start_job@ method. Use the footer@ and header@ methods to define the footer and header on a subsequent page before using the new_page@ method.

Header text has special formatting sequences. The sequence #p places the current page number in the header text. The sequence #P places the last page number in the header text. The sequence #dnnn places the date and time in the header text. The date and time used is the time at which the [start_job@](#) method is invoked. The string *nnn* is the format for the date, as defined in the datetim_.am file, located in the *install_dir/axdata/elf* directory. You do not need to include the datetim_.am file in your object method source to place a date and time in the header text.

See [PrinterClass Methods](#) for more information.

-> is_print_banner@

Sets banner page status

Class PrinterClass set

Format this.is_print_banner@(flag)

Arguments

flag	A Boolean value. TRUE sets banner page printing, FALSE disables banner page printing. The default value is the Print Banner Page(axPrintBanner) preference setting.
------	---

Description The set method `is_print_banner@` sets the banner page status. Use `is_print_banner@` to print a special page at the beginning of your document to help separate your document from other printer jobs.

See [PrinterClass Methods](#) for more information.

-> `is_print_collated@`

Sets collating status

Class PrinterClass set

Format `this.is_print_collated@(flag)`

Arguments flag A Boolean value. TRUE sets collating, FALSE disables collating. The default value is the Print Multiple Copies as Sets(`axPrintCollate`) preference setting.

Description The set method `is_print_collated@` sets the collating status. Use `is_print_collated@` to put multiple copies in order as sets, such as page 1,2,3,1,2,3 and so on. When collating is set, printing performance will be slower.

See [PrinterClass Methods](#) for more information.

-> `is_print_in_background@`

Sets background print status

Class PrinterClass set

Format `this.is_print_in_background@(flag)`

Arguments flag A Boolean value. TRUE sets background printing, FALSE disables background printing. The default value is the Print Document in the Background (`axPrintBackground`) preference setting.

Description The set method `is_print_in_background@` sets the background print status. Use `is_print_in_background@` to print information in the background, instead of waiting for the print job to finish before continuing application processing.

See [PrinterClass Methods](#) for more information.

-> is_print_in_color@

Sets color printer status

Class PrinterClass set

Format this.is_print_in_color@(flag)

Arguments flag A Boolean value. TRUE sets color printing, FALSE disables color printing. The default value is the Using Color Printer(axPrintIsColor) preference setting.

Description The set method is_print_in_color@ sets the collating status. Use is_print_in_color@ when you send print jobs to a color printer. If you choose this option and then send the document to a non-color printer, the colors are converted into grey scale values.

See [PrinterClass Methods](#) for more information.

-> new_page@

Starts new print page

Class PrinterClass set

Format this.new_page@

Description The set method new_page@ ends the current page and starts a new print page. Use the page_metrics@, footers@, and headers@ methods to set page, footer, and header information before using this method.

See [PrinterClass Methods](#) for more information.

-> page_dimensions@

Sets page dimensions

Class PrinterClass set

Format this.page_dimensions@(pageArray)

Arguments pageArray A two element array for the page dimensions. pageArray[0] is the page width, in mils (1000 mils = 1 inch). pageArray[1] is the page height, in mils.

Description The set method page_dimensions@ sets the print area page dimensions. Use this method to set custom page sizes, otherwise use the page_size@ method to set standard page sizes.

See [PrinterClass Methods](#) for more information.

-> page_margins@

Sets page margins

Class PrinterClass set

Format this.page_margins@(marginArray)

Arguments marginArray A four element array for the page margins, set to the following:

marginArray[0]	The left margin, in mils. (1000 mils = 1 inch)
marginArray[1]	The right margin, in mils.
marginArray[2]	The top margin, in mils.
marginArray[3]	The bottom margin, in mils.

Description The set method page_margins@ sets the print area page margins.

See [PrinterClass Methods](#) for more information.

-> page_metrics@

Sets page information

Class PrinterClass set

Format this.page_metrics@(size, orient, pageArray, marginArray)

Arguments size The standard page size. The standard page sizes are:

Letter	8500 mils W x 1100 mils H. (1000 mils = 1 inch)
Legal	8500 mils W x 1400 mils H.
Ledger	1700 mils W x 1100 mils H.

Executive	7500 mils W x 1000 mils H.
A3	11694 mils W x 16528 mils H.
A4	8264 mils W x 11694 mils H.
Custom	Page is set to a custom width and height with the set method <code>page_dimensions@</code> .
orient	The page orientation. Page orientation is either Portrait, printed across the shortest dimension of the page, or Landscape, printed across the longest dimension of the page.
pageArray	A two element array for the page dimensions. <code>pageArray[0]</code> is the page width, in mils. <code>pageArray[1]</code> is the page height, in mils. The <code>pageArray</code> is ignored if size is known
marginArray	A four element array for the page margins, set to the following:
<code>marginArray[0]</code>	The left margin, in mils.
<code>marginArray[1]</code>	The right margin, in mils.
<code>marginArray[2]</code>	The top margin, in mils.
<code>marginArray[3]</code>	The bottom margin, in mils.

Description The set method `page_metrics@` sets page information. You can use this method with different values for every page. Use the `page_metrics@`, `footers@`, and `headers@` methods to set page, footer, and header information before using the `new_page@` method.

See [PrinterClass Methods](#) for more information.

-> `page_orientation@`

Sets page orientation

Class PrinterClass set

Format `this.page_orientation@(orient)`

Arguments `orient` The page orientation. Page orientation is either Portrait, printed across the shortest dimension of the page, or Landscape, printed across the longest dimension of the page.

Description The set method `page_orientation@` sets page orientation. Use this method when you use the set method `page_size@` to set a standard page size.

See [PrinterClass Methods](#) for more information.

-> page_setup_dlg@

Displays a page setup dialog box

Class PrinterClass set

Format this.page_setup_dlg@

Description The set method page_setup_dlg@ displays a page setup dialog box. Use this method to display and set page attributes with a consistent graphical interface, instead of setting all page attributes individually.

See [PrinterClass Methods](#) for more information.

-> page_size@

Sets standard page size

Class PrinterClass set

Format this.page_size@(size)

Arguments

size	The standard page size. The defined standard page sizes are:
Letter	8500 mils W x 1100 mils H. (1000 mils = 1 inch)
Legal	8500 mils W x 1400 mils H.
Ledger	1700 mils W x 1100 mils H.
Executive	7500 mils W x 1000 mils H.
A3	11694 mils W x 16528 mils H.
A4	8264 mils W x 11694 mils H.

Description The set method page_size@ sets the print area to a standard page size.

See [PrinterClass Methods](#) for more information.

-> print@

Sends print job

Class PrinterClass set

Format this.print@

Description The set method print@ sends the print job to printer. Initialize the print job with start_job@ and set the printing parameters and information before using this method.

See [PrinterClass Methods](#) for more information.

-> printer_name@

Sets destination printer

Class PrinterClass set

Format this.printer_name@(destination)

Arguments destination The name of the destination printer. The default printer is the Set Default Printer Name(axPrintPrinter) preference setting.

Description The set method printer@ sets the destination printer. Set the printer name to NULL to print to file.

See [PrinterClass Methods](#) for more information.

-> printer_type@

Sets printer language

Class PrinterClass set

Format this.printer_type@(type)

Arguments type The string indicating the printer type, PostScript or PCL5. The default printer class is the Default Printer Type(axPrintDefaultType) preference setting.

Description The set method printer_type@ sets the printer type.

See [PrinterClass Methods](#) for more information.

-> print_file_name@

Sets printer file name

Class PrinterClass set

Format this.print_file_name@(file)

Arguments file The full path name of the print file.

Description The set method print_file_name@ sets the printer file name. If printer_name@ is set to NULL, the print job is printed to file, otherwise the file name is a temporary file spooled to the printer.

See [PrinterClass Methods](#) for more information.

-> print_setup_dlg@

Displays a print option dialog box

Class PrinterClass set

Format this.print_setup_dlg@

Description The set method print_setup_dlg@ displays a print option dialog box. Use this method to display print options in a consistent graphical interface, instead of setting all print options individually.

See [PrinterClass Methods](#) for more information.

-> reset@

Discards current print job

Class PrinterClass set

Format this.reset@

Description The set method reset@ discards the current print job

See [PrinterClass Methods](#) for more information.

-> start_job@

Initializes print job

Class PrinterClass set

Format this.start_job@([previewFlag[, title[, xpos[, ypos[, size]]]])

Arguments

previewFlag	A Boolean value. TRUE creates a preview canvas displayed on screen, and displays the print file in the preview canvas as it is created. FALSE does not create a preview screen. The default value is FALSE.
title	A text string that is the title of the preview canvas.
xpos	The x coordinate where the preview canvas is displayed on screen, relative to the mouse pointer position.
ypos	The y coordinate where the preview canvas is displayed on screen, relative to the mouse pointer position.
size	The size of the dialog box on screen. The passed integer is the percentage of the print page displayed. Valid values are from 25% to 100%. The default value is 50%.

Description The set method start_job@ initializes a print job and creates the first page of the print job file. Use the page_metrics@, footers@, and headers@ methods to set page, footer, and header information before using this method. Use the end_job@ method to send a print job file to a printer and end the print job.

See [PrinterClass Methods](#) for more information.

-> start_page@

Sets start page number

Class PrinterClass set

Format this.start_page@(num)

Arguments

num	The start page number. The start page must be an integer greater than or equal to 1. The default start page is 1.
-----	---

Description The set method start_page@ sets the start page number. The end_page@ and start_page@ methods determine the print job page range. If end_page@ is set to an integer less than start_page@, then the page range is all pages.

See [PrinterClass Methods](#) for more information.

<- error_event

Called for system errors

Class PrinterClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [PrinterClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class PrinterClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting the application main dialog box. This is a user-defined event, place all actions you want performed in the event definition, such as setting the printer defaults.

See [PrinterClass Methods](#) for more information.

-> new_page_event

Called before starting a new page

Class PrinterClass set

Format this.new_page_event(page)

Arguments page The page number.

Description The new_page_event is called by Applixware Builder before starting a new page in the print file. You can program a new_page_event to set different headers, footers, and page attributes for different pages. The new_page_event is given the page number as an argument. This is a user-defined event, place all actions you want performed in the event definition, such as setting the printer defaults.

For example, the following new_page_event places even page numbers in the left corner of the footer, while it places odd page numbers in the right corner of the footer.

```
set new_page_event(page_number)
    var object pen
    pen = this.child@("Pen")
    IF (page_number MOD 2 ) = 0
        this.footer@(pen,"#p",NULL,NULL,pen," ")
    ELSE
        this.footer@(pen," ",NULL,NULL,pen,"#p")
endset
```

See [PrinterClass Methods](#) for more information.

-> terminate_event

Called before application exit

Class PrinterClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before exiting the application. This is a user-defined event, place all actions you want performed before exiting the application in the event definition, such as saving or sending any started print jobs.

See [PrinterClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class PrinterClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [PrinterClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class RadioBoxClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the radio box. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color

- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- colorArray[1] The RGB red value.
- colorArray[2] The RGB blue value.
- colorArray[3] The RGB green value.

The following values apply to CMYK color type:

- colorArray[1] The CMYK cyan value.
- colorArray[2] The CMYK magenta value.
- colorArray[3] The CMYK yellow value.
- colorArray[4] The CMYK black value.

The following values apply to HSB color type:

- colorArray[1] The HSB hue value.
- colorArray[2] The HSB saturation value.
- colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [RadioBoxClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class RadioBoxClass get

Format colors = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears adjacent to the radio box choice.

The color settings are returned in the following format:

- colors[0] The color type. The valid color types are:
 - 1 RGB
 - 2 Named color
 - 3 Widget color
 - 4 Work area color

- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- colors[1] The RGB red value.
- colors[2] The RGB blue value.
- colors[3] The RGB green value.

The following values apply to CMYK color type:

- colors[1] The CMYK cyan value.
- colors[2] The CMYK magenta value.
- colors[3] The CMYK yellow value.
- colors[4] The CMYK black value.

The following values apply to HSB color type:

- colors[1] The HSB hue value.
- colors[2] The HSB saturation value.
- colors[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [RadioBoxClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class RadioBoxClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears adjacent to the radio box choice. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

- font_name The font name.
- font_size The font point size.
- bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [RadioBoxClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class RadioBoxClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. Use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears adjacent to the radio box choice.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [RadioBoxClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class RadioBoxClass get

Format flag = this.text_font_italic@

Description The get method text_font_italic@ returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. Use this method with the set method text_font_italic@ to verify and change the object text italic state.

Text appears adjacent to the radio box choice.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RadioBoxClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class RadioBoxClass get

Format name = this.text_font_name@

Description The get method text_font_name@ returns the object text font name. Use this method with the set method text_font_name@ to verify and change the object text font.

Text appears adjacent to the radio box choice.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"   'set method
```

See [RadioBoxClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class RadioBoxClass get

Format size = this.text_font_size@

Description The get method text_font_size@ returns the object text font size. Use this method with the set method text_font_size@ to verify and change the object text size.

Text appears adjacent to the radio box choice.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12        'set method
```

See [RadioBoxClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class RadioBoxClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears adjacent to the radio box choice. Text only appears shadowed on color displays.

See [RadioBoxClass Methods](#) for more information.

<- text_strings@

Returns radio box choices

Class RadioBoxClass get

Format valArray = this.text_strings@

Description The get method `text_strings@` returns the radio box choices as an array of strings. Use this method with the set method `text_strings@` to verify and change the radio box choices.

For example, to set the radio box choices use the following:

```
IF this.text_strings@ = NULL           'get method
    this.text_strings@ = {"red", "white", "blue"} 'set method
```

See [RadioBoxClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class RadioBoxClass get

Format colors = this.title_color@

Description The get method title_color@ returns a two dimensional array of title color settings.

Title is the text that appears above the radio box.

The color settings are returned in the following format:

colors[0]	The color type. The valid color types are:
1	RGB
2	Named color
3	Widget color
4	Work area color
5	HSB
6	CMYK

The following values apply to RGB color type:

colors[1]	The RGB red value.
colors[2]	The RGB blue value.
colors[3]	The RGB green value.

The following values apply to CMYK color type:

colors[1]	The CMYK cyan value.
colors[2]	The CMYK magenta value.
colors[3]	The CMYK yellow value.
colors[4]	The CMYK black value.

The following values apply to HSB color type:

colors[1]	The HSB hue value.
colors[2]	The HSB saturation value.
colors[3]	The HSB brightness value.

If the color type is a named color, then the color name is returned as colors[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [RadioBoxClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class RadioBoxClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [RadioBoxClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class RadioBoxClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears above the radio box.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [RadioBoxClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class RadioBoxClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears above the radio box.

For example, to make the object title italic use the following:

```
var flag
flag = this.title_font_italic@         'get method
IF NOT flag
    this.title_font_italic@ = TRUE     'set method
```

See [RadioBoxClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class RadioBoxClass get

Format name = this.title_font_name@

Description The get method title_font_name@ returns the object title font name. You can use this method with the set method title_font_name@ to verify and change the object text font.

Title is the text that appears above the radio box.

For example, to set the object title font to Courier use the following:

```
var name
name = this.title_font_name@           'get method
IF name <> "Courier"
    this.title_font_name@ = "Courier"   'set method
```

See [RadioBoxClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class RadioBoxClass get

Format size = this.title_font_size@

Description The get method title_font_size@ returns the object title font size. You can use this method with the set method title_font_size@ to verify and change the object text size.

Title is the text that appears above the radio box.

For example, to set the object title size to 12 use the following:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [RadioBoxClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class RadioBoxClass get

Format flag = this.title_is_shadowed@

Description The get method title_is_shadowed@ returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed,

otherwise it returns FALSE. Use this method with the set method `title_is_shadowed@` to verify and change the object text shadow state.

Title is the text that appears above the radio box choices. Text only appears shadowed on color displays.

See [RadioBoxClass Methods](#) for more information.

`<- value@`

Returns current radio box choice

Class RadioBoxClass get

Format `value = this.value@`

Description The get method `value@` returns the current radio box choice as a string. Use this method with the set method `text_strings@` to verify and change the radio box choices.

For example, to set the current radio box choice use the following:

```
IF this.value@ <> "blue"           'get method
    this.value@ = "blue"           'set method
```

See [RadioBoxClass Methods](#) for more information.

`<- value_index@`

Returns array index of radio box choice

Class RadioBoxClass get

Format `index = this.value_index@`

Description The get method `value_index@` returns the array index of the current radio box choice as a numeric value.

For example, to get the index of the current radio box choice use the following:

```
var values, index
values = this.text_strings@
index = this.value_index@
IF values[index] <> "blue"
    this.value@ = "blue"
```

See [RadioBoxClass Methods](#) for more information.

-> control_color@

Sets control color

Class RadioBoxClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method control_color@ sets the control color with an array of values.

See [RadioBoxClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class RadioBoxClass set

Format this.control_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method control_color_cmyk@ sets the control color with CMYK values.

See [RadioBoxClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class RadioBoxClass set

Format this.control_color_is_workspace@(flag)

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method control_color_is_workspace@ sets the color used by the object.

See [RadioBoxClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class RadioBoxClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method `control_color_name@` sets the control color by name.
See [RadioBoxClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class RadioBoxClass set

Format `this.control_color_rgb@(red, green, blue)`

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values.
See [RadioBoxClass Methods](#) for more information.

-> display@

Displays radio box

Class RadioBoxClass set

Format `this.display@`

Description The set method `display@` displays the radio box in the dialog box.
See [RadioBoxClass Methods](#) for more information.

-> text_color@

Sets text color

Class RadioBoxClass set

Format `this.text_color@(type, c1, c2, c3, c4)`

Arguments type

The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears adjacent to the radio box choice.

See [RadioBoxClass Methods](#) for more information.

-> text_color_cmyk@

Sets text CMYK color

Class RadioBoxClass set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments cyan The CMYK cyan value.

magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears adjacent to the radio box choice.

See [RadioBoxClass Methods](#) for more information.

-> `text_color_name@`

Sets text name color

Class RadioBoxClass set

Format `this.text_color_name@(name)`

Arguments name The color name.

Description The set method `text_color_name@` sets the text color by name. Text appears adjacent to the radio box choice.

See [RadioBoxClass Methods](#) for more information.

-> `text_color_rgb@`

Sets text RGB color

Class RadioBoxClass set

Format `this.text_color_rgb@(red, green, blue)`

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears adjacent to the radio box choice.

See [RadioBoxClass Methods](#) for more information.

-> text_font_attrs@

Sets text font information

Class RadioBoxClass set

Format this.text_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method text_font_attrs@ sets all the text font information. Text appears adjacent to the radio box choice. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format.

See [RadioBoxClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class RadioBoxClass set

Format this.text_font_bold@(flag)

Arguments

flag	Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.
------	---

Description The set method text_font_bold@ sets the text bold state. Text appears adjacent to the radio box choice. Use this method with the get method text_font_bold@ to verify and change the object text bold state. Use the set method text_font_attrs@ to set all font information in one step.

See [RadioBoxClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class RadioBoxClass set

Format this.text_font_italic@(flag)

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method text_font_italic@ sets the italic state of the object text. Text appears adjacent to the radio box choice. Use this method with the get method text_font_italic@ to verify and change the object text italic state. Use the set method text_font_attrs@ to set all font information in one step.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RadioBoxClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class RadioBoxClass set

Format this.text_font_name@(name)

Arguments name A string for the font.

Description The set method text_font_name@ returns the object text font name. Text appears adjacent to the radio box choice. Use this method with the get method text_font_name@ to verify and change the object text font. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text font to Courier use the following:

```
var name
```

```
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [RadioBoxClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class RadioBoxClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears adjacent to the radio box choice. Use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [RadioBoxClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class RadioBoxClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method `text_is_shadowed@` sets the shadow state of the object `text`. Use this method with the get method `text_is_shadowed@` to verify and change the object `text` shadow state.

Text appears adjacent to the radio box choice. Text only appears shadowed on color displays.

See [RadioBoxClass Methods](#) for more information.

-> text_strings@

Sets radio box choices

Class RadioBoxClass set

Format `this.text_strings@(valArray)`

Arguments `valArray` An array of strings for the radio box choices.

Description The set method `text_strings@` sets the radio box choices. Use this method with the get method `text_strings@` to verify and change the radio box choices.

For example, to set the radio box choices use the following:

<code>IF this.text_strings@ = NULL</code>	<code>'get method</code>
<code> this.text_strings@ = {"red", "white", "blue"}</code>	<code>'set method</code>

See [RadioBoxClass Methods](#) for more information.

-> title_color@

Sets title color

Class RadioBoxClass set

Format `this.title_color@(type, c1, c2, c3, c4)`

Arguments `type` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color

- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values. See [RadioBoxClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class RadioBoxClass set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values. See [RadioBoxClass Methods](#) for more information.

-> title_color_name@

Sets title name color

Class RadioBoxClass set

Format this.title_color_name@(name)

Arguments name The color name.

Description The set method title_color_name@ sets the title color by name.

See [RadioBoxClass Methods](#) for more information.

-> title_color_rgb@

Sets title RGB color

Class RadioBoxClass set

Format this.title_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [RadioBoxClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class RadioBoxClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments fonts The font information:
 font_name The font name.

font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `title_font_attrs@` sets all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [RadioBoxClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class RadioBoxClass set

Format `this.title_font_bold@(flag)`

Arguments flag Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.

Description The set method `title_font_bold@` sets the title bold state. You can use this method with the get method `title_font_bold@` to verify and change the object title bold state. Use the set method `title_font_attrs@` to set all font information in one step.

See [RadioBoxClass Methods](#) for more information.

-> title_font_italic@

Sets title italic state

Class RadioBoxClass set

Format `this.title_font_italic@(flag)`

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method `title_font_italic@` sets the italic state of the object title. You can use this method with the get method `title_font_italic@` to verify and change the object title italic state. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RadioBoxClass Methods](#) for more information.

-> title_font_name@

Sets title font name

Class RadioBoxClass set

Format `this.title_font_name@(name)`

Arguments name A string for the font.

Description The set method `title_font_name@` returns the object title font name. You can use this method with the get method `title_font_name@` to verify and change the object title font. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [RadioBoxClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class RadioBoxClass set

Format `this.title_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `title_font_size@` sets the object title font size. You can use this method with the get method `title_font_size@` to verify and change the object title size. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12        'set method
```

See [RadioBoxClass Methods](#) for more information.

-> `title_is_shadowed@`

Sets title shadow state

Class RadioBoxClass set

Format `this.title_is_shadowed@(flag)`

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method `title_is_shadowed@` sets the shadow state of the object title. Use this method with the get method `title_is_shadowed@` to verify and change the object title shadow state.

Title is the text that appears above the radio box choices. Text only appears shadowed on color displays.

See [RadioBoxClass Methods](#) for more information.

-> `value@`

Sets current radio box choice

Class RadioBoxClass set

Format `this.value@(value)`

Arguments value A string for the current radio box choice.

Description The set method `value@` sets the current radio box choice.

For example, to set the current radio box choice use the following:

```
IF this.value@ <> "blue"           'get method
    this.value@ = "blue"           'set method
```

See [RadioBoxClass Methods](#) for more information.

-> `value_index@`

Sets current radio box choice to array index

Class RadioBoxClass set

Format `this.value_index@(index)`

Arguments index A numeric value.

Description The set method `value_index@` sets the current radio box choice to the array string index of the passed value.

For example, to set the index of the current radio box choice to the array index of the string blue use the following:

```
var index, values
values = this.text_strings@
FOR index = 0 to ARRAY_SIZE@(values)-1
    IF values[index] = "blue"
        this.value_index@ = index
NEXT index
```

See [RadioBoxClass Methods](#) for more information.

<- `error_event`

Called for system errors

Class RadioBoxClass event

Format `flag = this.error_event(error_object)`

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [RadioBoxClass Methods](#) for more information.

-> changed_event

Called when radio box choice changes

Class RadioBoxClass event

Format this.changed_event(valueDex)

Arguments valueDex An integer indicating the menu choice index.

Description The changed_event is called by Applixware Builder when the the radio box choice changes. This is a user-defined event, place all actions you want performed in the event definition. The following is an example of a radio box changed_event.

```
set changed_event(value)
  IF value = 0
  {
    ' actions when radio box choice is 0
  }
  ELSE ' value is another radio box choice
  {
    ' actions when radio box choice is not 0
  }
```

See [RadioBoxClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class RadioBoxClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [RadioBoxClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class RadioBoxClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The set event resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [RadioBoxClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class RadioBoxClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [RadioBoxClass Methods](#) for more information.

-> `time_out_event`

Called on object timer time-out

Class `RadioBoxClass` event

Format `this.time_out_event`

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [RadioBoxClass Methods](#) for more information.

-> `update_event`

Called to update dialog box widget

Class `RadioBoxClass` event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box widget. This is a user-defined event, place all actions you want performed in the event definition.

See [RadioBoxClass Methods](#) for more information.

<- background_color@

Returns background colors used by object

Class RowColClass get

Format colorArray = this.background_color@

Description The get method background_color@ returns the background colors used by the Row-Col widgets. Colors appear only when a RowCol widget displays pixmaps. The method returns a two-dimensional array, with the color information set for each pixmap in the control. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [RowColClass Methods](#) for more information.

<- contents_type_is_pixmap@

Indicates the widget contents

Class RowColClass get

Format flag = this.contents_type_is_pixmap@

Description The get method contents_type_is_pixmap@ indicates the RowCol widget contents. The method returns TRUE if the widget contains pixmaps. The method returns FALSE if the widget contains strings.

See [RowColClass Methods](#) for more information.

<- contents_type_is_strings@

Indicates the widget contents

Class RowColClass get

Format flag = this.contents_type_is_strings@

Description The get method contents_type_is_strings@ indicates the RowCol widget contents. The method returns TRUE if the widget contains strings. The method returns FALSE if the widget contains pixmaps.

See [RowColClass Methods](#) for more information.

<- control_color@

Returns foreground colors used by object

Class RowColClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the foreground colors used by the RowCol widgets. Colors appear only when a RowCol widget displays pixmaps. The method returns a two-dimensional array, with the color information set for each pixmap in the control. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [RowColClass Methods](#) for more information.

<- is_drop_shadowed@

Returns RowCol widget shadowed state

Class RowColClass get

Format flag = this.is_drop_shadowed@

Description The get method is_drop_shadowed@ returns a Boolean value indicating the RowCol widget shadowed state. If the method returns TRUE the RowCol widget is shadowed, otherwise the method returns FALSE. A drop shadow gives a RowCol widget a 3-dimensional appearance.

See [RowColClass Methods](#) for more information.

<- number_of_columns@

Returns the quantity of widget columns

Class RowColClass get

Format flag = this.number_of_columns@

Description The get method `number_of_columns@` returns the quantity of RowCol widget columns. If the widget has a mult-column style, the information in the widget is displayed in the quantity of columns returned by this method. Use the get method `number_of_columns@` to change or set the quantity of columns in the widget.

See [RowColClass Methods](#) for more information.

<- pixmaps@

Returns the pixmaps used in the widget

Class RowColClass get

Format pixArray = this.pixmap@

Description The get method `pixmap@` returns an array of pixmaps used in the RowCol widget. Use the get method `pixmap@` to change or set the pixmaps in the widget.

See [RowColClass Methods](#) for more information.

<- pixmap_labels@

Returns the pixmap labels used in the widget

Class RowColClass get

Format strArray = this.pixmap_labels@

Description The get method `pixmap_labels@` returns the pixmap labels used in the RowCol widget as an array of strings. A string is displayed with the corresponding pixmap choice at the bottom of the RowCol widget. Use the set method [contents type is pixmaps@](#) to

set the widget content type for pixmaps, and set the pixmaps with the set method [pix-maps@](#) before assigning pixmap labels.

See [RowColClass Methods](#) for more information.

<- style_type@

Returns the widget display style

Class RowColClass get

Format styleNum = this.style_type@

Description The get method style_type@ returns a value indicating the RowCol widget display style. The valid styles are:

- 0 Square
- 1 Single column
- 2 Single row
- 3 Row major
- 4 Column major

Use the set method style_type@ to change or set the pixmaps in the widget.

See [RowColClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class RowColClass get

Format colorArray = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears in the RowCol widget.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color

- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- colorArray[1] The RGB red value.
- colorArray[2] The RGB blue value.
- colorArray[3] The RGB green value.

The following values apply to CMYK color type:

- colorArray[1] The CMYK cyan value.
- colorArray[2] The CMYK magenta value.
- colorArray[3] The CMYK yellow value.
- colorArray[4] The CMYK black value.

The following values apply to HSB color type:

- colorArray[1] The HSB hue value.
- colorArray[2] The HSB saturation value.
- colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [RowColClass Methods](#) for more information.

<- text_font_attrs@

Returns text font information

Class RowColClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method text_font_attrs@ returns all the text font information. Text appears in the RowCol widget. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

- font_name The font name.
- font_size The font point size.

bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [RowColClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class RowColClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. You can use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the RowCol widget.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [RowColClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class RowColClass get

Format flag = this.text_font_italic@

Description The get method text_font_italic@ returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it

returns FALSE. You can use this method with the set method `text_font_italic@` to verify and change the object text italic state.

Text appears in the RowCol widget.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RowColClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class RowColClass get

Format name = this.text_font_name@

Description The get method `text_font_name@` returns the object text font name. You can use this method with the set method `text_font_name@` to verify and change the object text font.

Text appears in the RowCol widget.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"   'set method
```

See [RowColClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class RowColClass get

Format size = this.text_font_size@

Description The get method `text_font_size@` returns the object text font size. You can use this method with the set method `text_font_size@` to verify and change the object text size. Text appears in the RowCol widget.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12        'set method
```

See [RowColClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class RowColClass get

Format flag = this.text_is_shadowed@

Description The get method `text_is_shadowed@` returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the RowCol widget. Text only appears shadowed on color displays.

See [RowColClass Methods](#) for more information.

<- text_strings@

Returns the strings used in the widget

Class RowColClass get

Format stringArray = this.text_strings@

Description The get method `text_strings@` returns an array of strings used in the RowCol widget. Use the set method `text_strings@` to change or set the strings in the widget.

See [RowColClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class RowColClass get

Format colorArray = this.title_color@

Description The get method title_color@ returns a two dimensional array of title color settings.

Title is the text that appears in the RowCol widget heading.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [RowColClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class RowColClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

See [RowColClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class RowColClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears in the RowCol widget heading.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
```

IF NOT flag

```
this.title_font_bold@ = TRUE
```

'set method

See [RowColClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class RowColClass get

Format flag = this.title_font_italic@

Description The get method `title_font_italic@` returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method `title_font_italic@` to verify and change the object title italic state.

Title is the text that appears in the RowCol widget heading.

For example, to make the object title italic use the following:

```
var flag
```

```
flag = this.title_font_italic@
```

'get method

```
IF NOT flag
```

```
this.title_font_italic@ = TRUE
```

'set method

See [RowColClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class RowColClass get

Format name = this.title_font_name@

Description The get method `title_font_name@` returns the object title font name. You can use this method with the set method `title_font_name@` to verify and change the object text font.

Title is the text that appears in the RowCol widget heading.

For example, to set the object title font to Courier use the following:

```
var name
```

```

name = this.title_font_name@           'get method
IF name <> "Courier"
    this.title_font_name@ = "Courier"   'set method

```

See [RowColClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class RowColClass get

Format size = this.title_font_size@

Description The get method `title_font_size@` returns the object title font size. You can use this method with the set method `title_font_size@` to verify and change the object text size.

Title is the text that appears in the RowCol widget heading.

For example, to set the object title size to 12 use the following:

```

var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method

```

See [RowColClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class RowColClass get

Format flag = this.title_is_shadowed@

Description The get method `title_is_shadowed@` returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. Use this method with the set method `title_is_shadowed@` to verify and change the object text shadow state.

Title is the text that appears in the RowCol widget heading. Text only appears shadowed on color displays.

See [RowColClass Methods](#) for more information.

<- value@

Returns current RowCol widget choice

Class RowColClass get

Format value = this.value@

Description The get method value@ returns the current RowCol widget choice as a string. You can use this method with the set method value@ to verify and change the RowCol widget choices.

For example, to set the current RowCol widget choice you would use the method as follows:

```
IF this.value@ <> "blue"           'get method
    this.value@ = "blue"           'set method
```

See [RowColClass Methods](#) for more information.

<- value_index@

Returns array index of RowCol widget choice

Class RowColClass get

Format index = this.value_index@

Description The get method value_index@ returns the array index of the current RowCol widget choice as a numeric value. The RowCol widget choices are a 0-based text string array.

For example, to get the index of the current RowCol widget choice you would use the method as follows:

```
var values, index
values = this.text_strings@
index = this.value_index@
IF values[index] <> "blue"
    this.value@ = "blue"
```

See [RowColClass Methods](#) for more information.

-> background_color@

Sets background color

Class RowColClass set

Format this.background_color@(type, c1, c2, c3, c4)

Arguments type An array of color types. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `background_color@` sets the background color with an array of values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> `background_color_cmyk@`

Sets background CMYK color

Class RowColClass set

Format `this.background_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `background_color_cmyk@` sets the background color with CMYK values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> `background_color_is_workspace@`

Sets color used by object

Class RowColClass set

Format `this.background_color_is_workspace@(flag)`

Arguments

<code>flag</code>	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
-------------------	---

Description The set method `background_color_is_workspace@` sets the color used by the object.

See [RowColClass Methods](#) for more information.

-> background_color_name@

Sets background name color

Class RowColClass set

Format this.background_color_name@(name)

Arguments name The color name.

Description The set method background_color_name@ sets the background color by name. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> background_color_rgb@

Sets background RGB color

Class RowColClass set

Format this.background_color_rgb@(red, green, blue)

Arguments red The RGB red value.
green The RGB blue value.
blue The RGB green value.

Description The set method background_color_rgb@ sets the background color with RGB values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> contents_type_is_pixmap@

Sets the widget content type

Class RowColClass set

Format this.contents_type_is_pixmap@(flag)

Arguments flag A Boolean variable. If flag is TRUE the RowCol widget contains pixmaps.
If flag is FALSE the RowCol widget contains strings.

Description The set method contents_type_is_pixmap@ sets the RowCol widget content type.
Use the set method pixmap@ to set the pixmap information.

See [RowColClass Methods](#) for more information.

-> contents_type_is_strings@

Sets the widget content type

Class RowColClass set

Format this.contents_type_is_strings@(flag)

Arguments flag A Boolean variable. If flag is TRUE the RowCol widget contains pixmaps.
If flag is FALSE the RowCol widget contains strings.

Description The set method contents_type_is_strings@ sets the RowCol widget content type. Use
the set method text_strings@ to set the string information.

See [RowColClass Methods](#) for more information.

-> control_color@

Sets foreground color

Class RowColClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type An array of color types. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the foreground color with an array of values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> control_color_cmyk@

Sets foreground CMYK color

Class RowColClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `control_color_cmyk@` sets the foreground color with CMYK values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the

method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class RowColClass set

Format this.control_color_is_workspace@(flag)

Arguments flag Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.

Description The set method control_color_is_workspace@ sets the foreground color used by the object.

See [RowColClass Methods](#) for more information.

-> control_color_name@

Sets foreground name color

Class RowColClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the foreground color by name. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> control_color_rgb@

Sets foreground RGB color

Class RowColClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the foreground color with RGB values. Colors appear only when a RowCol widget displays pixmaps. Each argument to the method must be an array of size greater than or equal to the quantity of pixmaps displayed in the control.

See [RowColClass Methods](#) for more information.

-> is_drop_shadowed@

Sets RowCol widget shadowed state

Class RowColClass set

Format this.is_drop_shadowed@(flag)

Arguments flag A Boolean value. TRUE makes the RowCol widget the drop shadowed.

Description The set method is_drop_shadowed@ sets the RowCol widget shadowed state. A drop shadow gives a RowCol widget a 3-dimensional appearance.

See [RowColClass Methods](#) for more information.

-> number_of_columns@

Sets the number of columns

Class RowColClass set

Format this.number_of_columns@(value)

Arguments value The number of columns.

Description The set method number_of_columns@ sets the number of columns displayed in the RowCol. Use the set method style@ to set a display style supporting multiple columns.

See [RowColClass Methods](#) for more information.

-> pixmaps@

Sets the pixmaps used in the widget

Class RowColClass set

Format this.pixmaps@(pixArray)

Arguments pixArray An array of pixmap names.

Description The set method `pixmaps@` sets the pixmaps used in the RowCol widget. Use the set method `contents_type_is_pixmaps@` to set the widget content type for pixmaps before assigning the array of pixmaps.

See [RowColClass Methods](#) for more information.

-> pixmap_labels@

Sets the pixmap labels used in the widget

Class RowColClass set

Format this.pixmap_labels@(strArray)

Arguments strArray An array of strings.

Description The set method `pixmap_labels@` sets the pixmap labels used in the RowCol widget. A string is displayed with the corresponding pixmap choice at the bottom of the RowCol widget. Use the set method [contents type is pixmaps@](#) to set the widget content type for pixmaps, and set the pixmaps with the set method [pixmaps@](#) before assigning the pixmap labels.

See [RowColClass Methods](#) for more information.

-> style_column_major@

Sets the widget display style

Class RowColClass set

Format this.style_column_major@

Description The set method `style_column_major@` sets the display style of the RowCol widget to fill columns first.

See [RowColClass Methods](#) for more information.

-> `style_row_major@`

Sets the widget display style

Class RowColClass set

Format `this.style_row_major@`

Description The set method `style_row_major@` sets the display style of the RowCol widget to fill rows first.

See [RowColClass Methods](#) for more information.

-> `style_single_column@`

Sets the widget display style

Class RowColClass set

Format `this.style_single_column@`

Description The set method `style_single_column@` sets the display style of the RowCol widget to display the information in a single column, similar to the RowCol widget.

See [RowColClass Methods](#) for more information.

-> `style_single_row@`

Sets the widget display style

Class RowColClass set

Format `this.style_single_row@`

Description The set method `style_single_row@` sets the display style of the RowCol widget to display the information in a single row.

See [RowColClass Methods](#) for more information.

-> style_square@

Sets the widget display style

Class RowColClass set

Format this.style_square@

Description The set method style_square@ sets the display style of the RowCol widget to display rows and columns equally.

See [RowColClass Methods](#) for more information.

-> style_type@

Sets the widget display style

Class RowColClass set

Format this.style_type@(type)

Arguments type A numeric value indicating the display style. The valid styles are:

- 0 Square
- 1 Single column
- 2 Single row
- 3 Row major
- 4 Column major

Description The set method style_type@ sets the display style of the RowCol widget.

See [RowColClass Methods](#) for more information.

-> text_color@

Sets text color

Class RowColClass set

Format `this.text_color@(type, c1, c2, c3, c4)`

Arguments `type` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- `c1` The RGB red value.
- `c2` The RGB blue value.
- `c3` The RGB green value.

The following values apply to CMYK color type:

- `c1` The CMYK cyan value.
- `c2` The CMYK magenta value.
- `c3` The CMYK yellow value.
- `c4` The CMYK black value.

The following values apply to HSB color type:

- `c1` The HSB hue value.
- `c2` The HSB saturation value.
- `c3` The HSB brightness value.

For a named color type, `c1` is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears in the RowCol widget.

See [RowColClass Methods](#) for more information.

-> text_color_cmyk@

Sets text CMYK color

Class RowColClass set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments cyan The CMYK cyan value.
 magenta The CMYK magenta value.
 yellow The CMYK yellow value.
 black The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears in the RowCol widget.

See [RowColClass Methods](#) for more information.

-> `text_color_name@`

Sets text name color

Class RowColClass set

Format `this.text_color_name@(name)`

Arguments name The color name.

Description The set method `text_color_name@` sets the text color by name. Text appears in the RowCol widget.

See [RowColClass Methods](#) for more information.

-> `text_color_rgb@`

Sets text RGB color

Class RowColClass set

Format `this.text_color_rgb@(red, green, blue)`

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the RowCol widget.

See [RowColClass Methods](#) for more information.

-> text_font_attrs@

Sets text font information

Class RowColClass set

Format this.text_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method text_font_attrs@ sets all the text font information. Text appears in the RowCol widget. The font_attrs_info@ format is defined in the *install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format.

See [RowColClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class RowColClass set

Format this.text_font_bold@(flag)

Arguments

flag	Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.
------	---

Description The set method text_font_bold@ sets the text bold state. Text appears in the RowCol widget. You can use this method with the get method text_font_bold@ to verify and change the object text bold state. Use the set method text_font_attrs@ to set all font information in one step.

See [RowColClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class RowColClass set

Format this.text_font_italic@(flag)

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method text_font_italic@ sets the italic state of the object text. Text appears in the RowCol widget. You can use this method with the get method text_font_italic@ to verify and change the object text italic state. Use the set method text_font_attrs@ to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RowColClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class RowColClass set

Format this.text_font_name@(name)

Arguments name A string for the font.

Description The set method text_font_name@ returns the object text font name. Text appears in the RowCol widget. You can use this method with the get method text_font_name@ to verify and change the object text font. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
```

```
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [RowColClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class RowColClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears in the RowCol widget. You can use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12 'set method
```

See [RowColClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class RowColClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method `text_is_shadowed@` sets the shadow state of the object text. Use this method with the get method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the RowCol widget. Text only appears shadowed on color displays.

See [RowColClass Methods](#) for more information.

-> `text_strings@`

Sets the strings used in the widget

Class RowColClass set

Format `this.text_strings@(strArray)`

Arguments strArray An array of strings.

Description The set method `text_strings@` sets the strings used in the RowCol widget. Use the set method `contents_type_strings@` to set the widget content type for strings.

See [RowColClass Methods](#) for more information.

-> `title_color@`

Sets title color

Class RowColClass set

Format `this.title_color@(type, c1, c2, c3, c4)`

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

c1 The RGB red value.

c2 The RGB blue value.

c3 The RGB green value.

The following values apply to CMYK color type:

c1 The CMYK cyan value.

c2 The CMYK magenta value.

c3 The CMYK yellow value.

c4 The CMYK black value.

The following values apply to HSB color type:

c1 The HSB hue value.

c2 The HSB saturation value.

c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values.

See [RowColClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class RowColClass set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values.

See [RowColClass Methods](#) for more information.

-> title_color_name@

Sets title name color

Class RowColClass set

Format this.title_color_name@(name)

Arguments name The color name.

Description The set method title_color_name@ sets the title color by name.

See [RowColClass Methods](#) for more information.

-> title_color_rgb@

Sets title RGB color

Class RowColClass set

Format this.title_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [RowColClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class RowColClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments fonts The font information:
 font_name The font name.

font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `title_font_attrs@` sets all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [RowColClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class RowColClass set

Format `this.title_font_bold@(flag)`

Arguments flag Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.

Description The set method `title_font_bold@` sets the title bold state. You can use this method with the get method `title_font_bold@` to verify and change the object title bold state. Use the set method `title_font_attrs@` to set all font information in one step.

See [RowColClass Methods](#) for more information.

-> title_font_italic@

Sets title italic state

Class RowColClass set

Format `this.title_font_italic@(flag)`

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method `title_font_italic@` sets the italic state of the object title. You can use this method with the get method `title_font_italic@` to verify and change the object title italic state. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [RowColClass Methods](#) for more information.

-> title_font_name@

Sets title font name

Class RowColClass set

Format `this.title_font_name@(name)`

Arguments name A string for the font.

Description The set method `title_font_name@` returns the object title font name. You can use this method with the get method `title_font_name@` to verify and change the object title font. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [RowColClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class RowColClass set

Format `this.title_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `title_font_size@` sets the object title font size. You can use this method with the get method `title_font_size@` to verify and change the object title size. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12        'set method
```

See [RowColClass Methods](#) for more information.

-> `title_is_shadowed@`

Sets title shadow state

Class RowColClass set

Format `this.title_is_shadowed@(flag)`

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method `title_is_shadowed@` sets the shadow state of the object title. Use this method with the get method `title_is_shadowed@` to verify and change the object title shadow state.

Title is the text that appears in the RowCol widget heading. Text only appears shadowed on color displays.

See [RowColClass Methods](#) for more information.

-> `value@`

Sets current RowCol widget choice

Class RowColClass set

Format `this.value@(value)`

Arguments value A string for the current RowCol widget choice.

Description The set method `value@` sets the current RowCol widget choice.

For example, to set the current RowCol widget choice you would use the method as follows:

```
IF this.value@ <> "blue"           'get method
    this.value@ = "blue"           'set method
```

See [RowColClass Methods](#) for more information.

-> `value_index@`

Sets current RowCol widget choice to array index

Class RowColClass set

Format `this.value_index@(index)`

Arguments index A numeric value.

Description The set method `value_index@` sets the current RowCol widget choice to the array string index of the passed value. The RowCol widget choices are a 0-based text string array.

For example, to set the index of the current RowCol widget choice to the array index of the string blue you would use the method as follows:

```
var index, values
values = this.text_strings@
FOR index = 0 to ARRAY_SIZE@(values)-1
    IF values[index] = "blue"
        this.value_index@ = index
NEXT index
```

See [RowColClass Methods](#) for more information.

-> `changed_event`

Called when a menu choice changes

Class RowColClass event

Format this.changed_event(value)

Arguments value The array index of the string or pixmap displayed in the RowCol widget.
 The array is 0-based.

Description The changed_event is called by Applixware Builder when the RowCol selection changes. This is a user-defined event, place all actions you want performed in the event definition. For example, the changed event for a RowCol widget may look as follows:

```
set changed_event(val)
  if val = 0
    ' Code for selecting first array item
  else
    ' Code for selecting other array items
endset
```

See [RowColClass Methods](#) for more information.

<- error_event

Called for system errors

Class RowColClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [RowColClass Methods](#) for more information.

-> initialize_event

Called before posting application dialog box

Class RowColClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before posting a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [RowColClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class RowColClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The resize_event is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [RowColClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class RowColClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [RowColClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class RowColClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [RowColClass Methods](#) for more information.

-> update_event

Called to update dialog box widget

Class RowColClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box widget. This is a user-defined event, place all actions you want performed in the event definition.

See [RowColClass Methods](#) for more information.

<- add_record@

Creates and returns a record

Class RealtimeGatewayClass get

Format record = this.add_record@(record, service, record_id, field_list, template)

Arguments

record	A string representing the record name
service	The logical name of the data feed within the data distribution system.
record_id	The name of the record of data in the data distribution system.
field_list	An array of field names for the record.
template	A defined template containing record field information. If a template is used the field_list argument should be passed a NULL value.

Description The get method add_record@ creates and returns a Real Time record. A record should be created as a child of a gateway object. See the set method [gateway@](#) for information on creating a Real Time gateway object.

See [RealtimeGatewayClass Methods](#) for more information.

<- add_template@

Creates and returns a template

Class RealtimeGatewayClass get

Format template = this.add_template@(template, field_list)

Arguments

template	A string representing the template name
field_list	An array of field names for the template.

Description The get method add_template@ creates and returns a template. A template contains an array of fields. You can use a template to define fields for multiple records, instead of defining individual fields and properties for each record.

See [RealtimeGatewayClass Methods](#) for more information.

<- gateway@

Returns a gateway name

Class RealtimeGatewayClass get

Format gateway = this.gateway@

Description The get method gateway@ returns a gateway name. A gateway is a Real Time engine. See [RealtimeGatewayClass Methods](#) for more information.

<- hostname@

Returns the host

Class RealtimeGatewayClass get

Format hostname = this.hostname@

Description The get method hostname@ returns a host machine name. The hostname can be set to NULL if the gateway runs locally.

See [RealtimeGatewayClass Methods](#) for more information.

<- is_connected@

Returns gateway connection status

Class RealtimeGatewayClass get

Format flag = this.is_connected@

Description The get method is_connected@ returns the gateway connection status. The method returns TRUE if a gateway connection exists, otherwise it returns FALSE.

See [RealtimeGatewayClass Methods](#) for more information.

<- record_children@

Returns gateway child records

Class RealtimeGatewayClass get

Format recordArray = this.record_children@

Description The get method record_children@ returns the records that are children of the gateway. The records are returned as an array of records.

See [RealtimeGatewayClass Methods](#) for more information.

<- rt_data_current_field_value@

Returns current field value

Class RealtimeGatewayClass get

Format value = this.rt_data_current_field_value@(record, field)

Arguments record The record name, as passed to the set method [register_record@](#).
field The field within the record.

Description The get method rt_data_current_field_value@ returns the current field value for a single field within a record.

See [RealtimeGatewayClass Methods](#) for more information.

<- template_children@

Returns gateway child templates

Class RealtimeGatewayClass get

Format templateArray = this.template_children@

Description The get method template_children@ returns the templates that are children of the gateway. The templates are returned as an array of templates.

See [RealtimeGatewayClass Methods](#) for more information.

-> add_record@

Creates a record

Class RealtimeGatewayClass set

Format this.add_record@(record, service, record_id, field_list, template)

Arguments

record	A string representing the record name
service	The logical name of the data feed within the data distribution system.
record_id	The name of the record of data in the data distribution system.
field_list	An array of field names for the record.
template	A defined template containing record field information. If a template is used the field_list argument should be passed a NULL value.

Description The set method add_record@ creates a Real Time record. A record should be created as a child of a gateway object. See the set method [gateway](#) for information on creating a Real Time gateway object.

See [RealtimeGatewayClass Methods](#) for more information.

-> add_template@

Creates a template

Class RealtimeGatewayClass set

Format this.add_template@(template, field_list)

Arguments

template	A string representing the template name
field_list	An array of field names for the template.

Description The set method add_template@ creates a template. A template contains an array of field names. You can use a template to define fields for multiple records, allowing global changes while updating only one template object.

See [RealtimeGatewayClass Methods](#) for more information.

-> connect@

Establishes gateway connection

Class RealtimeGatewayClass set

Format this.connect@

Description The set method connect@ establishes a gateway connection and registers all gateway records.

See [RealtimeGatewayClass Methods](#) for more information.

-> delete_record@

Deletes a record

Class RealtimeGatewayClass set

Format this.delete_record@(name)

Arguments name The record name, as created with the either the set or get method add_record.

Description The set method delete_record@ deletes records from the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> delete_template@

Deletes a record

Class RealtimeGatewayClass set

Format this.delete_template@(name)

Arguments name The template name, as created with the either the set or get method add_template.

Description The set method delete_template@ deletes templates from the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> disconnect@

Disconnects gateway

Class RealtimeGatewayClass set

Format this.disconnect@

Description The set method disconnect@ disconnects a gateway connection and unregisters all gateway records.

See [RealtimeGatewayClass Methods](#) for more information.

-> gateway@

Sets a gateway name

Class RealtimeGatewayClass set

Format this.gateway@(name)

Arguments name The name of the Real Time engine.

Description The get method gateway@ sets a gateway name. A gateway is a Real Time engine. If the gateway is already connected, the method will stop the current gateway, discard all records, and restart the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> hostname@

Disconnects gateway

Class RealtimeGatewayClass set

Format this.hostname@(name)

Arguments name The name of the host machine.

Description The set method hostname@ sets a host machine name. The hostname can be set to NULL if the gateway runs locally. If the gateway is already connected, the method will stop the current gateway, discard all records, and restart the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> register_all@

Registers all gateway records

Class RealtimeGatewayClass set

Format this.register_all@

Description The set method register_all@ registers all records that are children of the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> register_record@

Registers a gateway record

Class RealtimeGatewayClass set

Format this.register_record@(name)

Arguments name The record name, as created with the either the set or get method add_record@. The record should be a child of the gateway object.

Description The set method register_record@ registers a record that is a child of the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> unregister_all@

Unregisters all gateway records

Class RealtimeGatewayClass set

Format this.unregister_all@

Description The set method unregister_all@ unregisters all records that are children of the gateway.

See [RealtimeGatewayClass Methods](#) for more information.

-> unregister_record@

Unregisters a gateway record

Class RealtimeGatewayClass set

Format this.unregister_record@(name)

Arguments name The record name, as created with the either the set or get method add_record@. The record should be a child of the gateway object.

Description The set method unregister_record@ unregisters a record that is a child of the gateway. See [RealtimeGatewayClass Methods](#) for more information.

<- error_event

Called for system errors

Class RealtimeGatewayClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [RealtimeGatewayClass Methods](#) for more information.

-> initialize_event

Called when starting an application

Class RealtimeGatewayClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder to initialize RealtimeGatewayClass objects. This event is only called once, after the application's initialize_event. This is a user-defined event, place all actions to initialize the object attributes and system variables in the event definition.

See [RealtimeGatewayClass Methods](#) for more information.

-> terminate_event

Called before closing an application

Class RealtimeGatewayClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing an application. This event is only called once, before the application's terminate_event. This is a user-defined event, place all actions to free memory, reset attributes, and remove system variables in the event definition.

See [RealtimeGatewayClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class RealtimeGatewayClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is

set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The [timer@](#) method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [RealtimeGatewayClass Methods](#) for more information.

<- convert_publish_value@

Returns current record field published value

Class RealtimeRecordClass get

Format value = this.convert_publish_value@(field)

Arguments field A field within the record.

Description The get method `convert_publish_value@` returns the current value in the particular record field to be published to the Real Time data feed.

See [RealtimeRecordClass Methods](#) for more information.

<- convert_publish_value_by_dex@

Returns current record field publish value by index

Class RealtimeRecordClass get

Format valueArray = this.convert_publish_value_by_dex@(index)

Arguments index The field index position in the array of fields. Index positions are 0-based.

Description The get method `convert_publish_value_by_dex@` returns the current value of a record field to be published to the Real Time data feed by its index in the field array. Index positions are 0-based.

See [RealtimeRecordClass Methods](#) for more information.

<- current_value@

Returns current record field value

Class RealtimeRecordClass get

Format value = this.current_value@(field)

Arguments field A field within the record.

Description The get method current_value@ returns the current value in the particular record field.

See [RealtimeRecordClass Methods](#) for more information.

<- current_value_array@

Returns current record field values

Class RealtimeRecordClass get

Format valueArray = this.current_value_array@

Description The get method current_value_array@ returns the current value of all record fields.

See [RealtimeRecordClass Methods](#) for more information.

<- current_value_by_dex@

Returns current record field value by index

Class RealtimeRecordClass get

Format valueArray = this.current_value_by_dex@(index)

Arguments index The field index position in the array of fields. Index positions are 0-based.

Description The get method current_value_by_dex@ returns the current value of a record field by its index in the field array. Index positions are 0-based.

See [RealtimeRecordClass Methods](#) for more information.

<- display_value@

Returns displayed record field value

Class RealtimeRecordClass get

Format value = this.display_value@(field)

Arguments field A field within the record.

Description The get method display_value@ returns the display value in the particular record field. The display value may not be the same as the current value if the display field hasn't received updated information.

If a field converter macro or method is associated with the field the display value is the value returned by the field converter when the current value is passed to it.

See [RealtimeRecordClass Methods](#) for more information.

<- display_value_array@

Returns displayed record field values

Class RealtimeRecordClass get

Format valueArray = this.display_value_array

Description The get method display_value_array returns the display values of all record fields. The display values may not be the same as the current values if the display fields haven't received updated information.

See [RealtimeRecordClass Methods](#) for more information.

<- display_value_by_dex@

Returns displayed record field value by index

Class RealtimeRecordClass get

Format value = this.display_value_by_dex@(index)

Arguments index The field index position in the array of fields. Index positions are 0-based.

Description The get method `display_value@` returns the display value in the particular record field by its index in the field array. Index positions are 0-based. The display value may not be the same as the current value if the display field hasn't received updated information. If a field converter macro or method is associated with the field the display value is the value returned by the field converter when the current value is passed to it.

See [RealtimeRecordClass Methods](#) for more information.

<- field_converter@

Returns field converter macro

Class RealtimeRecordClass get

Format macro = this.field_converter@(field)

Arguments field A field within the record.

Description The get method `field_converter@` returns the field converter macro or method for the record field. The macro or method converts the current value to the display string.

See [RealtimeRecordClass Methods](#) for more information.

<- field_label@

Returns field label

Class RealtimeRecordClass get

Format label = this.field_label@(field)

Arguments field A field within the record.

Description The get method `field_label@` returns the field label for a record field.

See [RealtimeRecordClass Methods](#) for more information.

<- field_list@

Returns fields

Class RealtimeRecordClass get

Format format arrayof field_item@ fieldArray = this.field_list@

Description The get method field_list@ returns a record's fields as an array. Each array item is in the field_item@ format. The field_item@ format is defined in the builder_.am file, located in the *install_dir/axdata/elf* directory. The format is defined as:

```
format field_item@
    field_name,
    display_converter,
    field_label,
    publish_converter
```

See [RealtimeRecordClass Methods](#) for more information.

<- field_names_list@

Returns field names

Class RealtimeRecordClass get

Format fieldNameArray = this.field_names_list@

Description The get method field_names_list@ returns the field names used for a record. The field names are returned as an array of strings.

See [RealtimeRecordClass Methods](#) for more information.

<- field_publish_converter@

Returns field publish converter macro

Class RealtimeRecordClass get

Format macro = this.field_publish_converter@(field)

Arguments field A field within the record.

Description The get method `field_publish_converter@` returns the field publish converter macro or method for the record field. The macro or method converts the current value in the application to the value to be published to the Real Time data feed.

See [RealtimeRecordClass Methods](#) for more information.

<- field_publish_converter_by_dex@

Returns field publish converter macro by index

Class RealtimeRecordClass get

Format macro = this.field_publish_converter_by_dex@(index)

Arguments index The field index position in the array of fields. Index positions are 0-based.

Description The get method `field_publish_converter_by_dex@` returns the field publish converter macro or method for the record field by its index in the field array. Index positions are 0-based. The macro or method converts the current value in the application to the value to be published to the Real Time data feed.

See [RealtimeRecordClass Methods](#) for more information.

<- is_registered@

Returns registered status

Class RealtimeRecordClass get

Format flag = this.is_registered@

Description The get method `is_registered@` returns a Boolean value indicating the record registered status. The method returns TRUE if the record is registered with the gateway, otherwise it returns FALSE.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_status@

Returns field publish status

Class RealtimeRecordClass get

Format status = this.publish_status@(field)

Arguments field A field within the record.

Description The get method `publish_status@` returns a field publish status as a string. The method returns OK if the current value is published to the field, otherwise the method returns ERROR. Call this method in or after the `publish_status_event` to get the field status. The field status is maintained until the next publish event.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_status_array@

Returns field publish status of all record fields

Class RealtimeRecordClass get

Format statArray = this.publish_status_array@

Description The get method `publish_status_array@` returns field publish status of all fields in the record as an array of strings. The fields correspond to the fields returned by the get method `field_names_list@`. The method returns OK if the current value is published to the field, otherwise the method returns ERROR. Call this method in or after the `publish_status_event` to get the field status. The field status is maintained until the next publish event.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_status_by_dex@

Returns publish status of indexed field

Class RealtimeRecordClass get

Format status = this.publish_status_by_dex@(index)

Arguments index The field index in the record's array of fields.

Description The get method `publish_status_by_dex@` returns a field publish status as a string. The index position corresponds to the fields returned by the get method `field_names_list@`. The method returns OK if the current value is published to the field, otherwise the method returns ERROR. Call this method in or after the `publish_status_event` to get the field status. The field status is maintained until the next publish event.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_value@

Returns field value to be published

Class RealtimeRecordClass get

Format value = this.publish_value@(field)

Arguments field A field within the record.

Description The get method `publish_value@` returns a field value to be published.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_value_array@

Returns field values to be published for all record fields

Class RealtimeRecordClass get

Format valArray = this.publish_value_array@

Description The get method `publish_value_array@` returns field values to be published for all fields in the record as an array of values. The value indices correspond to the fields returned by the get method `field_names_list@`.

See [RealtimeRecordClass Methods](#) for more information.

<- publish_value_by_dex@

Returns value to be published for indexed field

Class RealtimeRecordClass get

Format value = this.publish_value_by_dex@(index)

Arguments index The field index in the record's array of fields.

Description The get method `publish_value_by_dex@` returns a field value to be published. The index position corresponds to the fields returned by the get method `field_names_list@`.

See [RealtimeRecordClass Methods](#) for more information.

<- record_id@

Returns record ID

Class RealtimeRecordClass get

Format record = this.record_id@

Description The get method `record_id@` returns the record ID. The record ID is used to register the record with the gateway and is the actual name used by the gateway to retrieve record information.

See [RealtimeRecordClass Methods](#) for more information.

<- record_label@

Returns record label

Class RealtimeRecordClass get

Format recordLabel = this.record_label@

Description The get method `record_label@` returns the record label. The record label provides a text string to refer to a record.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_database@

Returns database for RTSQL record

Class RealtimeRecordClass get

Format dbase = this.rtsql_database@

Description The get method rtsql_database@ returns the database used in an RTSQL record. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_host@

Returns host for database

Class RealtimeRecordClass get

Format host = this.rtsql_host@

Description The get method rtsql_host@ returns the name of the computer on which the database server is running, the database used for the RTSQL query. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_query@

Returns RTSQL query

Class RealtimeRecordClass get

Format query = this.rtsql_query@

Description The get method rtsql_query@ returns the SQL return query for the RTSQL record as an array of strings. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_routing@

Returns vendor gateway

Class RealtimeRecordClass get

Format gate = this.rtsql_routing@

Description The get method `rtsql_routing@` returns the vendor gateway name for the RTSQL record. The vendor gateway name is used if you are using more than one Oracle database. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_server@

Returns database server name

Class RealtimeRecordClass get

Format server = this.rtsql_server@

Description The get method `rtsql_server@` returns the database server name, if required by the database to establish the RTSQL connection. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_trigger@

Returns RTSQL trigger query

Class RealtimeRecordClass get

Format query = this.rtsql_trigger@

Description The get method `rtsql_trigger@` returns the SQL trigger query for the RTSQL record as a string or array of strings, depending on what is passed to the set method [rtsql_trigger@](#). A trigger query is a small, fast SQL query you use in the polling loop to

determine whether or not to execute the return query. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- rtsql_vendor@

Returns vendor name

Class RealtimeRecordClass get

Format vendor = this.rtsql_vendor@

Description The get method rtsql_vendor@ returns the database vendor name for the RTSQL record. The supported database vendors are Informix, Sybase, Oracle, and ODBC. The database names must be initial capitalized. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

<- service_name@

Returns service used by record

Class RealtimeRecordClass get

Format service = this.service_name@

Description The get method service_name@ returns the service used by the record. The service is the logical name of the data feed within the data distribution system.

See [RealtimeRecordClass Methods](#) for more information.

<- template_name@

Returns template used by record

Class RealtimeRecordClass get

Format template = this.template_name@

Description The get method `template_name@` returns the template used by the record. A template contains an array of fields. You can use a template to define fields for multiple records, instead of defining individual fields and properties for each record.

See [RealtimeRecordClass Methods](#) for more information.

-> `add_field_item@`

Adds a field to field list

Class RealtimeRecordClass set

Format `this.add_field_item@(name, converter, label)`

Arguments

<code>name</code>	The field name.
<code>converter</code>	The name of the macro used to convert the current value to the display string.
<code>label</code>	The label string used to reference the field.

Description The set method `add_field_item@` adds a field to the record field list. Use the method when you use individual fields instead of a template with the record.

See [RealtimeRecordClass Methods](#) for more information.

-> `delete_field_item@`

Deletes a field from field list

Class RealtimeRecordClass set

Format `this.delete_field_item@(name)`

Arguments

<code>name</code>	The field name.
-------------------	-----------------

Description The set method `delete_field_item@` deletes a field from the record field list. Use the method when you use individual fields instead of a template with the record.

See [RealtimeRecordClass Methods](#) for more information.

-> field_converter@

Sets a field converter

Class RealtimeRecordClass set

Format this.field_converter@(name, converter)

Arguments

name	The field name.
converter	The name of the macro or method used to convert the current value to the display string.

Description The set method field_converter@ sets a field converter macro or method. Use the method to convert values from the Real Time data feed for display in the application. Macro names must be preceded by the @ character, otherwise the converter is considered a get method.

See [RealtimeRecordClass Methods](#) for more information.

-> field_label@

Sets a field label

Class RealtimeRecordClass set

Format this.field_label@(name, label)

Arguments

name	The field name.
label	The label string used to reference the field.

Description The set method field_label@ sets a field label. Use the method when you use individual fields instead of a template with the record.

See [RealtimeRecordClass Methods](#) for more information.

-> field_list@

Sets record fields

Class RealtimeRecordClass set

Format this.field_list@(format arrayof field_item@ fieldArray)

Arguments fieldArray The array of fields. Each array item is in the field_item@ format. The field_item@ format is defined in the builder_.am file, located in the *install_dir/axdata/elf* directory. The format is defined as:

```
format field_item@
    field_name,
    display_converter,
    field_label,
    publish_converter
```

Description The set method field_list@ sets a record's fields. If a template is associated with the record, the method sets the template to NULL and uses the field list.

See [RealtimeRecordClass Methods](#) for more information.

-> field_names_list@

Sets record field names

Class RealtimeRecordClass set

Format this.field_names_list@(fieldNames)

Arguments fieldNames The array of field names.

Description The set method field_names_list@ sets a record's fields. If a template is associated with the record, the method sets the template to NULL and uses the field list.

See [RealtimeRecordClass Methods](#) for more information.

-> field_publish_converter@

Sets a field publish converter

Class RealtimeRecordClass set

Format this.field_publish_converter@(name, converter)

Arguments name The field name.

converter The name of the macro or method used to convert the the display string from the application to a published value.

Description The set method `field_publish_converter@` sets a field publish converter macro or method. Use the method to convert values from the application to a value published to the Real Time data feed. Macro names must be preceded by the `@` character, otherwise the converter is considered a get method.

See [RealtimeRecordClass Methods](#) for more information.

-> `publish@`

Publishes values to all record fields

Class `RealtimeRecordClass` set

Format `this.publish@(valArray)`

Arguments `valArray` An array of record field values. The array size should equal the number of fields in the record.

Description The set method `publish@` publishes an array of values to the gateway for all record fields. If no array is passed as an argument, the values set with the `publish_value@`, `publish_value_array@`, or `publish_value_by_dex@` methods are published to the gateway. The method flushes all values from all field registers.

See [RealtimeRecordClass Methods](#) for more information.

-> `publish_field@`

Publishes value for a record field

Class `RealtimeRecordClass` set

Format `this.publish_field@(field, value)`

Arguments `field` A record field.
`value` A field value to publish.

Description The set method `publish_field@` publishes a value to the gateway for the record field. If the passed field does not exist, an error is thrown. If no value is passed, the values set for the field with the `publish_value@`, `publish_value_array@`, or

`publish_value_by_dex@` methods are published to the gateway. The method flushes all values from the field register.

See [RealtimeRecordClass Methods](#) for more information.

-> `publish_fields@`

Publishes value for an array of record fields

Class RealtimeRecordClass set

Format `this.publish_fields@(fieldArray, valArray)`

Arguments `fieldArray` An array of record fields.
`valArray` An array of field values to publish.

Description The set method `publish_fields@` publishes an array of values to the gateway for the given record fields. If a passed field in `fieldArray` does not exist, an error is thrown. If no value is passed for a field, the values set for the field with the `publish_value@`, `publish_value_array@`, or `publish_value_by_dex@` methods are published to the gateway. The method flushes all values from the given field registers.

See [RealtimeRecordClass Methods](#) for more information.

-> `publish_fields_by_dex@`

Publishes value for an array of indexed fields

Class RealtimeRecordClass set

Format `this.publish_fields_by_dex@(dexArray, valArray)`

Arguments `dexArray` An array of record field indices.
`valArray` An array of field values to publish.

Description The set method `publish_fields_by_dex@` publishes an array of values to the gateway for the given record field indices. If a passed field index in `dexArray` does not exist, an error is thrown. If no value is passed for a field, the values set for the field with the `publish_value@`, `publish_value_array@`, or `publish_value_by_dex@` methods are published to the gateway. The index position corresponds to the fields returned by the get method `field_names_list@`. The method flushes all values from the given field registers.

See [RealtimeRecordClass Methods](#) for more information.

-> **publish_field_by_dex@**

Publishes value for an indexed field

Class RealtimeRecordClass set

Format this.publish_field_by_dex@(index, value)

Arguments

index	A record field index.
value	A field value to publish.

Description The set method `publish_field_by_dex@` publishes a value to the gateway for the given record field index. If the passed field index in `index` does not exist, an error is thrown. If no value is passed for a field, the values set for the field with the `publish_value@`, `publish_value_array@`, or `publish_value_by_dex@` methods are published to the gateway. The index position corresponds to the fields returned by the get method `field_names_list@`. The method flushes all values from the field register.

See [RealtimeRecordClass Methods](#) for more information.

-> **publish_value@**

Sets published value for a record field

Class RealtimeRecordClass set

Format this.publish_value@(field, value)

Arguments

field	A record field.
value	A field value to publish.

Description The set method `publish_value@` sets the published value for the given record field. The value is passed to the gateway when the `publish@` or any of the `publish_field@` methods are used with no value arguments. If the passed field does not exist, an error is thrown. The method stores all values in the field register.

See [RealtimeRecordClass Methods](#) for more information.

-> publish_value_array@

Sets published values for all record fields

Class RealtimeRecordClass set

Format this.publish_value_array@(valArray)

Arguments valArray An array of record field values. The array size should equal the number of fields in the record.

Description The set method publish_value_array@ sets the published values of all record fields. The value is passed to the gateway when the publish@ or any of the publish_field@ methods are used with no value arguments. The method stores all values in the field registers.

See [RealtimeRecordClass Methods](#) for more information.

-> publish_value_by_dex@

Sets published value for an indexed field

Class RealtimeRecordClass set

Format this.publish_value_by_dex@(index, value)

Arguments index A record field index.
value A field value to publish.

Description The set method publish_value_by_dex@ sets the published value for the given record field index. The value is passed to the gateway when the publish@ or any of the publish_field@ methods are used with no value arguments. The index position corresponds to the fields returned by the get method field_names_list@. The method stores all values in the field register.

See [RealtimeRecordClass Methods](#) for more information.

-> record_id@

Sets record ID

Class RealtimeRecordClass set

Format this.record_id@(record)

Arguments record The record ID.

Description The set method record_id@ sets the record ID. The record ID is used to register the record with the gateway and is the actual name used by the gateway to retrieve record information.

See [RealtimeRecordClass Methods](#) for more information.

-> record_label@

Sets record label

Class RealtimeRecordClass set

Format this.record_label@(label)

Arguments label The record label.

Description The set method record_label@ sets the record label. The record label provides a text string to refer to a record.

See [RealtimeRecordClass Methods](#) for more information.

-> register@

Registers a record

Class RealtimeRecordClass set

Format this.register@

Description The set method register @ registers a record with the parent gateway object, placing the record in the gateway's registered record table.

See [RealtimeRecordClass Methods](#) for more information.

-> rtsql_database@

Sets database for RTSQL record

Class RealtimeRecordClass set

Format this.rtsql_database@(dbase)

Arguments dbase The database name.

Description The set method rtsql_database@ sets the database used in an RTSQL record. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> rtsql_host@

Sets host for database

Class RealtimeRecordClass set

Format this.rtsql_host@(host)

Arguments host The name of the computer on which the database server is running.

Description The set method rtsql_host@ sets the name of the computer on which the database server is running, the database used for the RTSQL query. Use Real Time SQL for real time queries of a database through the rtsql gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> rtsql_query@

Sets RTSQL query

Class RealtimeRecordClass set

Format this.rtsql_query@(query)

Arguments query An array of strings containing the SQL query.

Description The set method `rtsql_host@` sets the SQL return query for the RTSQL record to the passes array of strings. Use Real Time SQL for real time queries of a database through the `rtsql` gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> `rtsql_routing@`

Sets vendor gateway

Class `RealtimeRecordClass` set

Format `this.rtsql_routing@(gate)`

Arguments `gate` The vendor gateway. Set the vendor gateway name if you are using more than one Oracle database.

Description The set method `rtsql_routing@` sets the vendor gateway name for the RTSQL record. The vendor gateway name is used if you are using more than one Oracle database. Use Real Time SQL for real time queries of a database through the `rtsql` gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> `rtsql_server@`

Sets database server name

Class `RealtimeRecordClass` set

Format `this.rtsql_server@(server)`

Arguments `server` The database server name.

Description The set method `rtsql_server@` sets the database server name, if required by the database to establish the RTSQL connection. Use Real Time SQL for real time queries of a database through the `rtsql` gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> `rtsql_trigger@`

Sets RTSQL trigger query

Class RealtimeRecordClass set

Format `this.rtsql_trigger@(query)`

Arguments query A string or array of strings containing the SQL query.

Description The set method `rtsql_trigger@` sets the SQL trigger query for the RTSQL record. A trigger query is a small, fast SQL query you use in the polling loop to determine whether or not to execute the return query. Use Real Time SQL for real time queries of a database through the `rtsql` gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> `rtsql_vendor@`

Sets database server name

Class RealtimeRecordClass set

Format `this.rtsql_vendor@(vendor)`

Arguments vendor The name of the database vendor. The supported database vendors are Informix, Sybase, Oracle, and ODBC. The database names must be initial capitalized.

Description The set method `rtsql_vendor@` sets the database vendor name for the RTSQL record. Use Real Time SQL for real time queries of a database through the `rtsql` gateway.

See [RealtimeRecordClass Methods](#) for more information.

-> `service_name@`

Sets service used by record

Class RealtimeRecordClass set

Format this.service_name@(service)

Arguments service The data feed service name.

Description The set method service_name@ sets the service used by the record. The service is the logical name of the data feed within the data distribution system.

See [RealtimeRecordClass Methods](#) for more information.

-> template_name@

Sets template used by record

Class RealtimeRecordClass set

Format this.template_name@(template)

Arguments template The field template name.

Description The set method template_name@ sets the service used by the record. A template contains an array of fields. You can use a template to define fields for multiple records, instead of defining individual fields and properties for each record.

See [RealtimeRecordClass Methods](#) for more information.

-> unregister@

Unregisters a record

Class RealtimeRecordClass set

Format this.unregister@

Description The set method unregister @unregisters a record with the parent gateway object, removing the record from the gateway's registered record table.

See [RealtimeRecordClass Methods](#) for more information.

-> use_template@

Sets template used by record

Class RealtimeRecordClass set

Format this.use_template@(template)

Arguments template The field template name.

Description The set method use_template@ sets the service used by the record. A template contains an array of fields. You can use a template to define fields for multiple records, instead of defining individual fields and properties for each record.

See [RealtimeRecordClass Methods](#) for more information.

-> data_update_event

Called when a field value changes

Class RealtimeRecordClass event

Format this.data_update_event(fieldArray)

Arguments fieldArray An array of field indexes. The indexes are 0-based.

Description The data_update_event is called by Applixware Builder when a field value changes. The indexes of the changes fields are passed as an array to the event. Use the event to track changes in designated fields. This is a user-defined event, place all actions you want performed in the event definition.

See [RealtimeRecordClass Methods](#) for more information.

<- error_event

Called for system errors

Class RealtimeRecordClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [RealtimeRecordClass Methods](#) for more information.

-> initialize_event

Called when starting an application

Class RealtimeRecordClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder to initialize RealtimeRecordClass objects. This event is only called once, after the application's initialize_event. This is a user-defined event, place all actions to initialize the object attributes and system variables in the event definition.

See [RealtimeRecordClass Methods](#) for more information.

-> publish_status_event

Called when a field value is published

Class RealtimeRecordClass event

Format this.publish_status_event(fieldArray)

Arguments fieldArray An array of field indexes. The indexes are 0-based.

Description The publish_status_event is called by Applixware Builder when a field value is published. The indexes of the changes fields are passed as an array to the event. Use the event with the publish_status@ methods to determine successful or unsuccessful

publishing actions. This is a user-defined event, place all actions you want performed in the event definition.

See [RealtimeRecordClass Methods](#) for more information.

-> terminate_event

Called before closing an application

Class RealtimeRecordClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing an application. This event is only called once, before the application's terminate_event. This is a user-defined event, place all actions to free memory, reset attributes, and remove system variables in the event definition.

See [RealtimeRecordClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class RealtimeRecordClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [RealtimeRecordClass Methods](#) for more information.

<- field_converter@

Returns field converter macro

Class RealtimeTemplateClass get

Format macro = this.field_converter@(field)

Arguments field A field within the template.

Description The get method field_converter@ returns the field converter macro for the template field. The macro converts the current value to the display string.

See [RealtimeTemplateClass Methods](#) for more information.

<- field_label@

Returns field label

Class RealtimeTemplateClass get

Format label = this.field_label@(field)

Arguments field A field within the record.

Description The get method field_label@ returns the field label for a template field.

See [RealtimeTemplateClass Methods](#) for more information.

<- field_list@

Returns fields

Class RealtimeTemplateClass get

Format format arrayof field_item@ fieldArray = this.field_list@

Description The get method field_list@ returns a template's fields as an array. Each array item is in the field_item@ format. The field_item@ format is defined in the builder_.am file, located in the *install_dir/axdata/elf* directory. The format is defined as:

format field_item@

field_name,
display_converter,
field_label,
publish_converter

See [RealtimeTemplateClass Methods](#) for more information.

<- field_names_list@

Returns field names

Class RealtimeTemplateClass get

Format fieldNameArray = this.field_names_list@

Description The get method field_names_list@ returns the field names used for a template. The field names are returned as an array of strings.

See [RealtimeTemplateClass Methods](#) for more information.

<- field_publish_converter@

Returns field publish converter macro

Class RealtimeTemplateClass get

Format macro = this.field_publish_converter@(field)

Arguments field A field within the record.

Description The get method field_publish_converter@ returns the field publish converter macro or method for the record field. The macro or method converts the current value in the application to the value to be published to the Real Time data feed.

See [RealtimeTemplateClass Methods](#) for more information.

-> add_field_item@

Adds a field to field list

Class RealtimeTemplateClass set

Format this.add_field_item@(name, converter, label)

Arguments

name	The field name.
converter	The name of the macro used to convert the current value to the display string.
label	The label string used to reference the field.

Description The set method add_field_item@ adds a field to the template field list.

See [RealtimeTemplateClass Methods](#) for more information.

-> delete_field_item@

Deletes a field from field list

Class RealtimeTemplateClass set

Format this.delete_field_item@(name)

Arguments

name	The field name.
------	-----------------

Description The set method delete_field_item@ deletes a field from the template field list.

See [RealtimeTemplateClass Methods](#) for more information.

-> field_converter@

Sets a field converter

Class RealtimeTemplateClass set

Format this.field_converter@(name, converter)

Arguments

name	The field name.
converter	The name of the macro used to convert the current value to the display string.

Description The set method field_converter@ sets a field converter macro.

See [RealtimeTemplateClass Methods](#) for more information.

-> field_label@

Sets a field label

Class RealtimeTemplateClass set

Format this.field_label@(name, label)

Arguments name The field name.
 label The label string used to reference the field.

Description The set method field_label@ sets a field label.
See [RealtimeTemplateClass Methods](#) for more information.

-> field_list@

Sets template fields

Class RealtimeTemplateClass set

Format this.field_list@(format arrayof field_item@ fieldArray)

Arguments fieldArray The array of fields. Each array item is in the field_item@ format. The field_item@ format is defined in the builder_.am file, located in the *install_dir/axdata/elf* directory. The format is defined as:

format field_item@
 field_name,
 display_converter,
 field_label,
 publish_converter

Description The set method field_list@ sets a template's fields.
See [RealtimeTemplateClass Methods](#) for more information.

-> field_names_list@

Sets template field names

Class RealtimeTemplateClass set

Format this.field_names_list@(fieldNames)

Arguments fieldNames The array of field names.

Description The set method field_names_list@ sets a template's fields.

See [RealtimeTemplateClass Methods](#) for more information.

-> field_publish_converter@

Sets a field publish converter

Class RealtimeTemplateClass set

Format this.field_publish_converter@(name, converter)

Arguments name The field name.

converter The name of the macro or method used to convert the the display string from the application to a published value.

Description The set method field_publish_converter@ sets a field publish converter macro or method. Use the method to convert values from the application to a value published to the Real Time data feed. Macro names must be preceded by the @ character, otherwise the converter is considered a get method.

See [RealtimeTemplateClass Methods](#) for more information.

<- error_event

Called for system errors

Class RealtimeTemplateClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [RealtimeTemplateClass Methods](#) for more information.

-> initialize_event

Called when starting an application

Class RealtimeTemplateClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder to initialize RealtimeTemplateClass objects. This event is only called once, after the application's initialize_event. This is a user-defined event, place all actions to initialize the object attributes and system variables in the event definition.

See [RealtimeTemplateClass Methods](#) for more information.

-> terminate_event

Called before closing an application

Class RealtimeTemplateClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing an application. This event is only called once, before the application's terminate_event. This is a user-defined event, place all actions to free memory, reset attributes, and remove system variables in the event definition.

See [RealtimeTemplateClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class RealtimeTemplateClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [RealtimeTemplateClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class ScalerClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the scale. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB

6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [ScalerClass Methods](#) for more information.

<- height@

Returns scale height

Class ScalerClass get

Format pixels = this.height@

Description The get method height@ returns the scale height in pixels. Use the method with the set method height@ to verify and change the object's height.

For example, to make the scale height at least 10 pixels use the method as follows:

```
var pixels
```

```
pixels = this.height@           'get method
```

```
IF pixels < 10
```

```
    this.height@ = 10           'set method
```

See [ScalerClass Methods](#) for more information.

<- max_value@

Returns maximum scale value

Class ScalerClass get

Format value = this.max_value@

Description The get method max_value@ returns the maximum scale value. Use the method with the set method max_value@ to verify and change the scale's maximum value.

For example, to make the maximum scale value at least 100 use the method as follows:

```
var value
value = this.max_value@           'get method
IF value < 100
    this.max_value@ = 100        'set method
```

See [ScalerClass Methods](#) for more information.

<- min_value@

Returns minimum scale value

Class ScalerClass get

Format value = this.min_value@

Description The get method min_value@ returns the minimum scale value. Use the method with the set method min_value@ to verify and change the scale's minimum value.

For example, to make the minimum scale value at least 5 use the method as follows:

```
var value
value = this.min_value@         'get method
IF value < 5
    this.min_value@ = 5        'set method
```

See [ScalerClass Methods](#) for more information.

<- value@

Returns the current scale value

Class ScalerClass get

Format value = this.value@

Description The get method value@ returns the current scale value. Use the method with the set method value@ to verify and change the scale's current value.

For example, to set the current scale value to 25 use the method as follows:

```
var value
value = this.value@           'get method
IF value < 25
    this.value@ = 25         'set method
```

See [ScalerClass Methods](#) for more information.

<- width@

Returns scale width

Class ScalerClass get

Format pixels = this.width@

Description The get method width@ returns the scale width in pixels. Use the method with the set method width@ to verify and change the object's width.

For example, to make the scale width at least 25 pixels use the method as follows:

```
var pixels
pixels = this.width@         'get method
IF pixels < 25
    this.width@ = 25         'set method
```

See [ScalerClass Methods](#) for more information.

-> control_color@

Sets control color

Class ScalerClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method control_color@ sets the control color with an array of values.

See [ScalerClass Methods](#) for more information.

-> control_color_cmyk@

Sets control CMYK color

Class ScalerClass set

Format this.control_color_cmyk@(cyan, magenta, yellow, black)

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method control_color_cmyk@ sets the control color with CMYK values. See [ScalerClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class ScalerClass set

Format this.control_color_is_workspace@(flag)

Arguments

flag	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
------	---

Description The set method control_color_is_workspace@ sets the color used by the object. See [ScalerClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class ScalerClass set

Format this.control_color_name@(name)

Arguments

name	The color name.
------	-----------------

Description The set method `control_color_name@` sets the control color by name.

See [ScalerClass Methods](#) for more information.

-> `control_color_rgb@`

Sets control RGB color

Class ScalerClass set

Format `this.control_color_rgb@(red, green, blue)`

Arguments

red	The RGB red value.
green	The RGB blue value.
blue	The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values.

See [ScalerClass Methods](#) for more information.

-> `height@`

Sets scale height

Class ScalerClass set

Format `this.height@(value)`

Arguments value The height, in pixels, of the scale.

Description The set method `height@` sets the scales's height. Use the method with the get method `height@` to verify and change the object's height.

For example, to make the scale height at least 10 pixels use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 10
    this.height@ = 10          'set method
```

See [ScalerClass Methods](#) for more information.

-> max_value@

Sets maximum scale value

Class ScalerClass set

Format this.max_value@(value)

Arguments value The maximum value of the scale.

Description The set method max_value@ sets the scale's maximum value. Use the method with the get method max_value@ to verify and change the scale's maximum value.

For example, to make the maximum scale value at least 100 use the method as follows:

```
var value
value = this.max_value@           'get method
IF value < 100
    this.max_value@ = 100        'set method
```

See [ScalerClass Methods](#) for more information.

-> min_value@

Sets minimum scale value

Class ScalerClass set

Format this.min_value@(value)

Arguments value The minimum value of the scale.

Description The set method min_value@ sets the scale's minimum value. Use the method with the get method min_value@ to verify and change the scale's minimum value.

For example, to make the minimum scale value at least 5 use the method as follows:

```
var value
value = this.min_value@           'get method
IF value < 5
    this.min_value@ = 5          'set method
```

See [ScalerClass Methods](#) for more information.

-> value@

Sets current scale value

Class ScalerClass set

Format this.value@(value)

Arguments value The current value of the scale.

Description The set method value@ sets the scale's current value. The current value must be between the minimum and maximum values, inclusive. Use the method with the get method value@ to verify and change the scale's current value.

For example, to set the current scale value to 25 use the method as follows:

```
var value
value = this.value@      'get method
IF value < > 25
    this.value@ = 25    'set method
```

See [ScalerClass Methods](#) for more information.

-> width@

Sets scale width

Class ScalerClass set

Format this.width@(value)

Arguments value The width, in pixels, of the scale.

Description The set method width@ sets the scale's width. Use the method with the get method width@ to verify and change the object's width.

For example, to make the scale width at least 25 pixels use the method as follows:

```
var pixels
pixels = this.width@    'get method
IF pixels < 25
    this.width@ = 25    'set method
```

See [ScalerClass Methods](#) for more information.

-> **changed_event**

Called when scale value changes

Class ScalerClass event

Format this.changed_event(value)

Arguments value The current value of the scale.

Description The changed_event is called by Applixware Builder when the scale value changes. This is a user-defined event, place all actions you want performed in the event definition.

See [ScalerClass Methods](#) for more information.

<- **error_event**

Called for system errors

Class ScalerClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ScalerClass Methods](#) for more information.

-> initialize_event

Called before displaying a dialog box control

Class ScalerClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the scale dimensions, position, and so on.

See [ScalerClass Methods](#) for more information.

-> motion_event

Called when scale drag action occurs

Class ScalerClass event

Format this.motion_event(value)

Arguments value The current value of the scale.

Description The motion_event is called by Applixware Builder when a scale drag action occurs. This is a user-defined event, place all actions you want performed in the event definition.

For example, to have a label in the dialog box display the value of the scale after a motion_event use the following:

```
set motion_event(newval)
```

```
  var object Lab
```

```
  Lab = this.sibling@("Label")
```

```
  Lab.value@ = newval
```

```
endset
```

See [ScalerClass Methods](#) for more information.

-> **resize_event**

Called when a dialog box is resized

Class ScalerClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [ScalerClass Methods](#) for more information.

-> **terminate_event**

Called before closing or destroying dialog box

Class ScalerClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ScalerClass Methods](#) for more information.

-> **time_out_event**

Called on object timer time-out

Class ScalerClass event

Format this.time_out_event

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the `timer@` method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [ScalerClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class ScalerClass event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [ScalerClass Methods](#) for more information.

<- marker_width@

Returns splitter width

Class SplitterClass get

Format `pixels = this.marker_width@`

Description The get method `marker_width@` returns the splitter width in pixels. The minimum splitter width is 4 pixels. The default splitter width is 16.

See [SplitterClass Methods](#) for more information.

<- max_value@

Returns maximum splitter value

Class SplitterClass get

Format value = this.max_value@

Description The get method max_value@ returns the maximum splitter value. The maximum value determines the splitter's working height in the dialog box. The minimum value is 0, set at the splitter's initial y-position in the dialog box. Use the method with the set method max_value@ to verify and change the splitter's maximum value.

For example, to make the maximum splitter value at least 100 use the method as follows:

```
var value
value = this.max_value@      'get method
IF value < 100
    this.max_value@ = 100    'set method
```

See [SplitterClass Methods](#) for more information.

<- value@

Returns the current splitter value

Class SplitterClass get

Format value = this.value@

Description The get method value@ returns the current splitter value. The splitter value must be between the minimum value, 0, and the maximum value set with max_value@. Use the method with the set method value@ to verify and change the splitter's current value.

For example, to set the current splitter value to 25 use the method as follows:

```
var value
value = this.value@          'get method
IF value < >25
    this.value@ = 25         'set method
```

See [SplitterClass Methods](#) for more information.

-> marker_width@

Sets splitter height

Class SplitterClass set

Format this.marker_width@(value)

Arguments value The width, in pixels, of the splitter.

Description The set method marker_width@ sets the splitters's width. The minimum splitter width is 4 pixels. The default splitter width is 16.

See [SplitterClass Methods](#) for more information.

-> max_value@

Sets maximum splitter value

Class SplitterClass set

Format this.max_value@(value)

Arguments value The maximum value of the splitter.

Description The set method max_value@ sets the maximum splitter value. The maximum value determines the splitter's working height in the dialog box. The minimum value is 0, set at the splitter's initial y-position in the dialog box. Use the method with the set method max_value@ to verify and change the splitter's maximum value.

For example, to make the maximum splitter value at least 100 use the method as follows:

```
var value
value = this.max_value@      'get method
IF value < 100
    this.max_value@ = 100    'set method
```

See [SplitterClass Methods](#) for more information.

-> value@

Sets current splitter value

Class SplitterClass set

Format this.value@(value)

Arguments value The current value of the splitter.

Description The set method value@ sets the splitter's current value. The splitter value must be between the minimum value, 0, and the maximum value set with max_value@. Use the method with the get method value@ to verify and change the splitter's current value.

For example, to set the current splitter value to 25 use the method as follows:

```
var value
value = this.value@      'get method
IF value < > 25
    this.value@ = 25    'set method
```

See [SplitterClass Methods](#) for more information.

<- error_event

Called for system errors

Class SplitterClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [SplitterClass Methods](#) for more information.

-> **changed_event**

Called when splitter value changes

Class SplitterClass event

Format this.changed_event(value)

Arguments value The current value of the splitter.

Description The changed_event is called by Applixware Builder when the splitter value changes. This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

-> **initialize_event**

Called before displaying a control

Class SplitterClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the splitter size, position, and so on.

See [SplitterClass Methods](#) for more information.

-> **motion_event**

Called when splitter drag action occurs

Class SplitterClass event

Format this.motion_event(value)

Arguments value The current position of the splitter.

Description The motion_event is called by Applixware Builder when a splitter drag action occurs. This event is called This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

-> picked_event

Called when splitter is picked

Class SplitterClass event

Format this.picked_event(value)

Arguments value The current position of the splitter.

Description The picked_event is called by Applixware Builder when the splitter is picked with a left mouse button press. This event is called before the first splitter motion_event. This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

-> released_event

Called when splitter is released

Class SplitterClass event

Format this.released_event(value)

Arguments value The current position of the splitter.

Description The released_event is called by Applixware Builder when a left mouse button press is released from the splitter. This event is called after the final splitter motion_event. This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

-> **resize_event**

Called when a dialog box is resized

Class SplitterClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [SplitterClass Methods](#) for more information.

-> **terminate_event**

Called before closing or destroying dialog box

Class SplitterClass event

Format this.terminate_event

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

-> **time_out_event**

Called on object timer time-out

Class SplitterClass event

Format this.time_out_event

Description The `time_out_event` is called by Applixware Builder on an object timer time-out. A `time_out_event` is generated when the object's timer expires. The time out interval is set with the `timer@` method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The `timer@` method would be used to set the time interval to 1 second. A `time_out_event` would occur after 1 second.

See [SplitterClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class SplitterClass event

Format `this.update_event`

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [SplitterClass Methods](#) for more information.

<- column@

Returns value of column in current row

Class SQLCommandClass get

Format `val = this.column@(num)`

Arguments `num` The column number. Column numbers are 0-based.

Description The get method `column@` returns the value of the column in the current database row.

See [SQLCommandClass Methods](#) for more information.

<- cursor_status@

Returns value of column in current row

Class SQLCommandClass get

Format format cursor_status@ stat = this.cursor_status@

Description The get method cursor_status@ returns the cursor status in cursor_status@ format. The format is defined in the *install_dir/axdata/elf/sqlc_.am* header file. The format is defined as:

format cursor_status@

open,	'TRUE if cursor is open
editingEnabled,	'TRUE if cursor is opened for editing
implicitLock,	'TRUE if rows are locked when values are changed, 'false if lock_row@ must be called explicitly
transactionState,	'has a transaction started on this connection? EDIT#OFF 0 'editing is not happening EDIT#TRANSACTION 1 'an edit has occurred in the dbms, a transaction in underway EDIT#TRANSACTIONLESS 2 'an edit has occurred, but dbms does not have transactions
unAppliedEdits,	'TRUE if edits have been made but not applied, note that requery will clear unapplied edits.
AppliedEdits	'TRUE if there are applied edits, note that requery will clear this flag.

See [SQLCommandClass Methods](#) for more information.

<- description@

Returns table and column information

Class SQLCommandClass get

Format format sql_description@ col = this.description@

Description The get method `description@` returns database table information in `sql_description@` format. The format contains the following information:
format arrayof `sql_table_desc@` tables,
format arrayof `sql_column_desc@` columns
The `sql_description@` format is defined in the `dbase_.am` file. Data types are defined in the `dbase_.am` file located in the `install_dir/axdata/elf` directory.
See [SQLCommandClass Methods](#) for more information.

<- display_binary_data@

Displays a BLOB in corresponding editor, returns handle to editor

Class SQLCommandClass get

Format editor = this.display_binary_data@(blob)

Arguments blob A binary large object.

Description The get method `display_binary_data@` displays a BLOB in the appropriate editor for the object and returns a handle to the editor object. This method only works with the ODBC gateway. Use the get method [get_binary_data@](#) to retrieve BLOB data from the database.

See [SQLCommandClass Methods](#) for more information.

<- fetchsize@

Returns quantity of rows to fetch

Class SQLCommandClass get

Format size = this.fetchsize@

Description The get method `fetchsize@` returns the quantity of rows to fetch at one time.

See [SQLCommandClass Methods](#) for more information.

<- get_binary_data@

Returns a BLOB

Class SQLCommandClass get

Format blob = this.get_binary_data@(col, nameOnlyFlag)

Arguments col The column number. Column numbers are 0-based.
nameOnlyFlag A Boolean value, TRUE means only return the file name of the BLOB, FALSE means return the actual BLOB data.

Description The get method `get_binary_data@` returns BLOB data for the specified column in the current row. This method only works with the ODBC gateway. Use the get method [display_binary_data@](#) to display the BLOB in the appropriate editor.

See [SQLCommandClass Methods](#) for more information.

<- is_all_rows_fetched@

Indicates if all rows have been fetched

Class SQLCommandClass get

Format flag = this.is_all_rows_fetched@

Description The get method `is_all_rows_fetched@` returns a Boolean value indicating if all rows have been fetched. If the method returns TRUE all rows meeting the query conditions have been fetched, otherwise the method returns FALSE.

See [SQLCommandClass Methods](#) for more information.

<- is_column_updatable@

Returns column update status

Class SQLCommandClass get

Format flag = this.is_column_updatable@(num)

Arguments num The column number. Column numbers are 0-based.

Description The get method `is_column_updatable@` returns a Boolean value indicating if the column in the current row is updatable. The method returns TRUE if the row is updatable, otherwise the method returns FALSE.

See [SQLCommandClass Methods](#) for more information.

`<- is_row_locked@`

Returns row lock status

Class SQLCommandClass get

Format flag = this.is_row_locked@

Description The get method `is_row_locked@` returns the row lock status as a Boolean value. The method returns TRUE if the current row is locked, otherwise it returns FALSE.

See [SQLCommandClass Methods](#) for more information.

`<- is_statement_editable@`

Returns editable status of current prepared statement

Class SQLCommandClass get

Format flag = this.is_statement_editable@

Description The get method `is_statement_editable@` returns the editable status of current prepared statement as a Boolean value. The method returns TRUE if a cursor can be opened for editing on the prepared statement, otherwise it returns FALSE.

See [SQLCommandClass Methods](#) for more information.

`<- next_pos@`

Returns next row number

Class SQLCommandClass get

Format num = this.next_pos@

Description The get method `next_pos@` returns the number of the next row. The method returns NULL if there are no more rows. The method returns the first row number if there is no current row.

See [SQLCommandClass Methods](#) for more information.

`<- odbc_fetch_rowset@`

Returns rows of data from a result set

Class SQLCommandClass get

Format `format sql_result@ data = this.odbc_fetch_rowset@(row, headFlag)`

Arguments

<code>row</code>	The row number.
<code>headFlag</code>	A Boolean value, TRUE means include the column heading information with the returned data.

Description The get method `odbc_fetch_rowset@` returns a rowset of data from the result set

See [SQLCommandClass Methods](#) for more information.

`<- pos@`

Returns current row number

Class SQLCommandClass get

Format `num = this.pos@`

Description The get method `pos@` returns the number of the current row. The method returns NULL if there is no current row.

See [SQLCommandClass Methods](#) for more information.

`<- row@`

Returns current row info

Class SQLCommandClass get

Format dataArray = this.row@

Description The get method row@ returns the current row information in an array. The array contains any edits to the row.

See [SQLCommandClass Methods](#) for more information.

<- rows_fetched@

Returns quantity of rows fetched

Class SQLCommandClass get

Format qty= this.rows_fetched@

Description The get method rows_fetched@ returns the quantity of rows fetched by the query.

See [SQLCommandClass Methods](#) for more information.

<- row_status@

Returns row status

Class SQLCommandClass get

Format status= this.row_status@

Description The get method row_status@ returns the row status. The row status is one of the following values:

1 Inserted

2 Edited

3 Deleted

NULL No change

See [SQLCommandClass Methods](#) for more information.

-> apply_edits@

Applies edits to database

Class SQLCommandClass set

Format this.apply_edits@

Description The set method apply_edits@ applies edits to the database.

See [SQLCommandClass Methods](#) for more information.

-> close_cursor@

Closes cursor to database

Class SQLCommandClass set

Format this.close_cursor@

Description The set method close_cursor@ closes the cursor to the database.

See [SQLCommandClass Methods](#) for more information.

-> commit@

Commits edits

Class SQLCommandClass set

Format this.commit@

Description The set method commit@ commits database edits and closes the transaction. The edits must be applied to the table before committing. Once you commit edits, changes are made to the database. You cannot use rollback@ to undo the edits after they are committed.

See [SQLCommandClass Methods](#) for more information.

-> delete_row@

Marks a row for deletion

Class SQLCommandClass set

Format this.delete_row@

Description The set method delete_row@ marks a row for deletion, but does not change the row numbers of the current query.

See [SQLCommandClass Methods](#) for more information.

-> display_binary_data@

Displays a BLOB in corresponding editor

Class SQLCommandClass set

Format this.display_binary_data@(blob)

Arguments blob A binary large object.

Description The set method display_binary_data@ displays a BLOB in the appropriate editor for the object. This method only works with the ODBC gateway. Use the get method [get_binary_data@](#) to retrieve BLOB data from the database.

See [SQLCommandClass Methods](#) for more information.

-> fetch_all_rows@

Fetches all rows

Class SQLCommandClass set

Format this.fetch_all_rows@

Description The set method fetch_all_rows@ fetches all rows from the database that meet the query condition.

See [SQLCommandClass Methods](#) for more information.

-> insert_row@

Inserts a row

Class SQLCommandClass set

Format this.insert_row@

Description The set method insert_row@ inserts a row before the current row. If there is now current row, the row is inserted at the end of all rows.

See [SQLCommandClass Methods](#) for more information.

-> lock_row@

Locks a row

Class SQLCommandClass set

Format this.lock_row@

Description The set method lock_row@ attempts to lock the current row. You cannot use this method to lock inserted or deleted rows.

See [SQLCommandClass Methods](#) for more information.

-> next_pos@

Makes next row current

Class SQLCommandClass set

Format this.next_pos@

Description The set method next_pos@ makes the next row the current row. If there is no current row, the method makes the first row the current row.

See [SQLCommandClass Methods](#) for more information.

-> open_cursor@

Opens a cursor

Class SQLCommandClass set

Format this.open_cursor@(fetchSize, editFlag, lockFlag)

Arguments

fetchSize	The number of rows to retrieve at one time.
editFlag	A Boolean value, TRUE means open the cursor for editing.
lockFlag	A Boolean value, TRUE means lock a row when any column in the row is updated.

Description The set method open_cursor@ opens a cursor in the result set.

See [SQLCommandClass Methods](#) for more information.

-> pos@

Sets current row

Class SQLCommandClass set

Format this.pos@(num)

Arguments

num	A row number. Row numbers are 0-based. Set num to NULL to have no current row.
-----	--

Description The set method pos@ sets the current row.

See [SQLCommandClass Methods](#) for more information.

-> prepare@

Prepares SQL statement for query

Class SQLCommandClass set

Format this.prepare@(sql, editFlag[, fetchSize[, keyCols]])

Arguments

sql	The SQL string.
-----	-----------------

editFlag	A Boolean value, set to TRUE to enable editing of the query results.
fetchSize	An optional argument, sets the number of rows to retrieve in a query.
keyCols	An optional argument that is used when editFlag is TRUE. This argument is an array of hidden key column names that may be required for editing.

Description The set method `prepare@` prepares an SQL statement for a database query. See [SQLCommandClass Methods](#) for more information.

-> `put_binary_data@`

Inserts a BLOB in a column

Class SQLCommandClass set

Format `this.put_binary_data@(blob)`

Arguments

col	The column number in the row. Column numbers are 0-based.
blob	A binary large object.

Description The set method `put_binary_data@` inserts a BLOB in the specified column of the current row. The column must support binary data types. This method only works with the ODBC gateway. Use the get method [get_binary_data@](#) to retrieve BLOB data from the database.

See [SQLCommandClass Methods](#) for more information.

-> `rollback@`

Cancels database edits

Class SQLCommandClass set

Format `this.rollback@`

Description The set method `rollback@` cancels all edits made against the current query. The method cannot undo edits that are committed to the database.

See [SQLCommandClass Methods](#) for more information.

-> unedit_row@

Remove row edits

Class SQLCommandClass set

Format this.unedit_row@

Description The set method unedit_row@ cancels all edits made in a row, returning the row to its previous state. If you use this method with an inserted row, the inserted row is removed.

See [SQLCommandClass Methods](#) for more information.

-> unprepare@

Unprepares database

Class SQLCommandClass set

Format this.unprepare@

Description The set method unprepare@ unprepares a database.

See [SQLCommandClass Methods](#) for more information.

-> update_column@

Updates column in current row

Class SQLCommandClass set

Format this.update_column@(colNum, value)

Arguments colNum The column number. Column numbers are 0-based.
value The new column value.

Description The set method update_column@ updates a column in the current row with the passed value.

See [SQLCommandClass Methods](#) for more information.

<- channel@

Returns channel

Class SQLConnectClass get

Format format sql_channel@ channels = this.channel@

Description The get method channel@ returns the database connection channel in sql_channel@ format. The sql_channel@ format is defined in the dbase_.am file located in the install_dir/axdata/elf directory.

See [SQLConnectClass Methods](#) for more information.

<- command@

Creates and returns a SQLCommandClass object

Class SQLConnectClass get

Format object comm = this.command@

Description The get method command@ creates and initializes an SQLCommandClass object, then returns a reference to the object.

See [SQLConnectClass Methods](#) for more information.

<- database@

Returns database name

Class SQLConnectClass get

Format dbase = this.database@

Description The get method database@ returns the name of the currently connected database as a string.

See [SQLConnectClass Methods](#) for more information.

<- exec_direct@

Executes SQL statement, returns results

Class SQLConnectClass get

Format result = this.exec_direct@(sql, autoFlag)

Arguments

sql	An SQL string.
autoFlag	A Boolean value, which only affects connections through the ODBC gateway. If the arguments is TRUE, the results of the statement are automatically committed to the database. If the argument is FALSE then the results are not committed to the database.

Description The get method exec_direct@ executes the passed SQL statement and returns the results.

See [SQLConnectClass Methods](#) for more information.

<- host@

Returns database host name

Class SQLConnectClass get

Format host = this.host@

Description The get method host@ returns the name of the database host machine.

See [SQLConnectClass Methods](#) for more information.

<- is_transactionless@

Returns database transaction status

Class SQLConnectClass get

Format flag = this.is_transactionless@

Description The get method `is_transactionless@` returns the database transaction status. The method returns TRUE if the database does not have transactions. The method returns FALSE if the database has transactions.

See [SQLConnectClass Methods](#) for more information.

<- server@

Returns database server

Class SQLConnectClass get

Format server = this.server@

Description The get method `server@` returns the name of the database server.

See [SQLConnectClass Methods](#) for more information.

<- transaction_state@

Returns transaction state

Class SQLConnectClass get

Format state = this.transaction_state@

Description The get method `transaction_state@` returns the transaction state of the database server. The state is indicated by the following return values:

0	EDIT#OFF	No edit is underway.
1	EDIT#TRANSACTION	An edit has occurred, a transaction is in progress.
2	EDIT#TRANSACTIONLESS	An edit has occurred, but database does not have transactions.

See [SQLConnectClass Methods](#) for more information.

<- vendor@

Returns database vendor

Class SQLConnectClass get

Format vendor = this.vendor@

Description The get method vendor@ returns the name of the database vendor.

See [SQLConnectClass Methods](#) for more information.

-> close_gateway@

Closes gateway connection

Class SQLConnectClass set

Format this.close_gateway@

Description The set method close_gateway@ close the database gateway connection.

See [SQLConnectClass Methods](#) for more information.

-> commit@

Commits edits

Class SQLConnectClass set

Format this.commit@

Description The set method commit@ commits database edits and closes the transaction. You cannot use rollback@ to undo the edits after they are committed.

See [SQLConnectClass Methods](#) for more information.

-> connect@

Establishes connection to database server

Class SQLConnectClass set

Format this.connect@(server, database[, user[, pwd[, format sql_cursor_properties@ cursor]])

Arguments server The database server name.

 database The database name.

user	The user name.
pwd	The user password.
cursor	The cursor properties, in sql_cursor_properties@ format. The format is defined in the <i>install_dir/axdata/elf/dbase_.am</i> header file.

Description The set method connect@ establishes a connection to a database server.

See [SQLConnectClass Methods](#) for more information.

-> disconnect@

Disconnects from database server

Class SQLConnectClass set

Format this.disconnect@

Description The set method disconnect@ disconnects from the database server but does not close the gateway connection.

See [SQLConnectClass Methods](#) for more information.

-> exec_direct@

Executes SQL statemen

Class SQLConnectClass set

Format this.exec_direct@(sql, autoFlag)

Arguments

sql	An SQL string.
autoFlag	A Boolean value, which only affects connections through the ODBC gateway. If the arguments is TRUE, the results of the statement are automatically committed to the database. If the argument is FALSE then the results are not committed to the database.

Description The set method exec_direct@ executes the passed SQL statement. Use the get method [exec_direct@](#) if you need to execute an SQL statement and return the results.

See [SQLConnectClass Methods](#) for more information.

-> open_gateway@

Starts gateway process

Class SQLConnectClass set

Format this.open_gateway@(host, gate)

Arguments

host	The gateway host machine name.
gate	The vendor gateway. The following strings are for Applixware supplied gateways: Informix Oracle Sybase ODBC

If gate is not an Applixware database gateway, the method assumes that gate is a custom gateway and attempts to start the custom gateway.

Description The set method open_gateway@ starts a gateway process.

See [SQLConnectClass Methods](#) for more information.

-> rollback@

Cancels database edits

Class SQLConnectClass set

Format this.rollback@

Description The set method rollback@ cancels all edits made against the current query. The method cannot undo edits that are committed to the database.

See [SQLConnectClass Methods](#) for more information.

<- active_layer@

Returns active layer name

Class TabControlClass get

Format name = this.active_layer@

Description The get method active_layer@ returns the active layer name.

See [TabControlClass Methods](#) for more information.

<- containees@

Returns contents of a layer

Class TabControlClass get

Format object ctls = this.containees@([name])

Arguments name The layer name.

Description The get method containees@ returns an array of references to the controls in the named layer. If the layer name is not supplied, references to all controls in the tab control are returned.

See [TabControlClass Methods](#) for more information.

<- layernames@

Returns layer names

Class TabControlClass get

Format nameArray = this.layernames@

Description The get method layernames@ returns the layer names as an array of strings.

See [TabControlClass Methods](#) for more information.

<- top_inset@

Returns tab height

Class TabControlClass get

Format height = this.top_inset@

Description The get method top_inset@ returns the height, in pixels, of the tab area of the control. The tab area contains the label, and is the area you click on to change the selected layer.

See [TabControlClass Methods](#) for more information.

-> active_layer@

Sets active layer

Class TabControlClass set

Format this.active_layer@(name)

Arguments name The layer name.

Description The set method active_layer@ sets the active layer of the tab by layer name.

See [TabControlClass Methods](#) for more information.

-> insert_control@

Adds a control to a layer

Class TabControlClass set

Format this.insert_control@(name, object insert)

Arguments name The layer name.
insert The object to insert in the layer.

Description The set method insert_control@ adds a control to a layer in the tab control. You must still set the coordinates of the control so that it appears within the layer.

See [TabControlClass Methods](#) for more information.

-> **layernames@**

Sets layer names

Class TabControlClass set

Format this.layernames@(nameArray)

Arguments name An array of layer names.

Description The set method active_layer@ sets the layer names for the tab control.

See [TabControlClass Methods](#) for more information.

-> **load_dbox@**

Inserts a dialog box in a layer

Class TabControlClass set

Format this.load_dbox@(name, dbox)

Arguments name The layer name.
dbox The dialog box file name. If the dialog box is not in your ELF search path you must supply the full path name.

Description The set method load_dbox@ inserts the contents of a dialog box file in a tab layer. Design your dialog box so that it will fit the tab layer dimensions. You will still need to program the object method source of controls added to the layer.

See [TabControlClass Methods](#) for more information.

<- **error_event**

Called for system errors

Class TabControlClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The error_event is called by Applixware Builder for posting object errors. If an error_event for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an error_event for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [TabControlClass Methods](#) for more information.

-> initialize_event

Called before displaying a dialog box

Class TabControlClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the panel dimensions, color, and so on.

See [TabControlClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class TabControlClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments

width	The changed dialog box width.
height	The changed dialog box height.
old_width	The original dialog box width.
old_height	The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition, such as repositioning or resizing controls in the dialog box.

See [TabControlClass Methods](#) for more information.

-> `tab_event`

Called when a tab is selected

Class `TabControlClass` event

Format `this.tab_event(layer)`

Arguments `layer` The layer name.

Description The `tab_event` is called by Applixware Builder when a tab is selected. This is a user-defined event, place all actions you want performed in the event definition. For example, you can program the event to load controls dynamically on a layer with the `load_dbox@` method, or to set the state of a control on a layer.

See [TabControlClass Methods](#) for more information.

-> `terminate_event`

Called before closing or destroying dialog box

Class `TabControlClass` event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [TabControlClass Methods](#) for more information.

-> `time_out_event`

Called on object timer time-out

Class `TabControlClass` event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [TabControlClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class TabControlClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [TabControlClass Methods](#) for more information.

<- cell_editing_is_allowed@

Returns the table cell edit status

Class TableClass get

Format flag = this.cell_editing_is_allowed@

Description The get method cell_editing_is_allowed@ returns the table cell edit status. The method returns TRUE if table cell editing is allowed, otherwise the method returns FALSE. When table cells are editable, you can click in a cell and type at the text cursor.

For example, to enable table cell editing use the following:

```
if this.cell_editing_is_allowed@ = FALSE      'get method
```

`this.cell_editing_is_allowed@ = TRUE` 'set method

See [TableClass Methods](#) for more information.

<- column_resizing_is_allowed@

Returns the column resizing status

Class TableClass get

Format `flag = this.column_resizing_is_allowed@`

Description The get method `column_resizing_is_allowed@` returns the column resizing status as a Boolean value. The method returns TRUE if the table allows column resizing. The method returns FALSE if the table columns are a fixed size and do not allow resizing.

See [TableClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class TableClass get

Format `colorArray = this.control_color@`

Description The get method `control_color@` returns the color used by the table. The array information is returned as follows:

`colorArray[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

`colorArray[1]` The RGB red value.

`colorArray[2]` The RGB blue value.

`colorArray[3]` The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [TableClass Methods](#) for more information.

<- data_set_field_name_list@

Retruns data set fields used in the table

Class TableClass get

Format fieldArray = this.data_set_field_name_list@

Description The get method data_set_field_name_list@ returns the data set fields displayed in the table. If data set field names are used the [heading_info@](#) is NULL.

See [TableClass Methods](#) for more information.

<- display_lines@

Returns height of table in lines

Class TableClass get

Format val = this.display_lines@

Description The get method display_lines@ returns the height of the table as the number of displayed lines in the table.

See [TableClass Methods](#) for more information.

<- heading_info@

Returns table headings

Class TableClass get

Format valArray = this.heading_info@

Description The get method heading_info@ returns a two-dimensional array of heading information. The array has the following format:

valArray[x,0] The column heading string.

valArray[x,1] The column width, in pixels.

Use this method with the set method heading_info@ to verify and change the table heading information.

See [TableClass Methods](#) for more information.

<- height@

Returns table height

Class TableClass get

Format pixels = this.height@

Description The get method height@ returns the table height in pixels. You can use this method with the set method height@ to verify and change the object's height.

For example, to make the table height at least 100 pixels you would use the method as follows:

```
var pixels
```

```
pixels = this.height@           'get method
```

```
IF pixels < 100
```

```
    this.height@ = 100         'set method
```

See [TableClass Methods](#) for more information.

<- horizontal_grid_is_hidden@

Returns the horizontal grid line status

Class TableClass get

Format flag = this.horizontal_grid_is_hidden@

Description The get method horizontal_grid_is_hidden@ returns the horizontal grid line status as a Boolean value. The method returns TRUE if the horizontal grid lines are hidden. The method returns FALSE if the horizontal grid lines are displayed.

See [TableClass Methods](#) for more information.

<- hscroll_is_enabled@

Returns horizontal scroll bar status

Class TableClass get

Format flag = this.hscroll_is_enabled@

Description The get method hscroll_is_enabled@ returns the horizontal scroll bar status as a Boolean value. The method returns TRUE if the horizontal scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

See [TableClass Methods](#) for more information.

<- hscroll_length@

Returns horizontal scroll bar length

Class TableClass get

Format pixels = this.hscroll_length@

Description The get method hscroll_length@ returns the horizontal scroll bar length, in pixels.

See [TableClass Methods](#) for more information.

<- hscroll_origin@

Returns horizontal scroll bar origin

Class TableClass get

Format pixels = this.hscroll_origin@

Description The get method `hscroll_origin@` returns the horizontal scroll bar origin, in pixels, from the bottom left corner of the table.

See [TableClass Methods](#) for more information.

<- is_editable@

Returns the table edit status

Class TableClass get

Format flag = this.is_editable@

Description The get method `is_editable@` returns the table edit status. The method returns TRUE if the table is editable, otherwise it returns FALSE. Use this method with the set method `is_editable@` to verify and change the table edit status.

For example, to enable table editing use the following:

```
if this.is_editable@ = FALSE           'get method
    this.is_editable@ = TRUE 'set method
```

See [TableClass Methods](#) for more information.

<- is_multi_select@

Returns table multiple row selection status

Class TableClass get

Format flag = this.is_multi_select@

Description The get method `is_multi_select@` returns the table multiple row selection status as a Boolean value. The method returns TRUE if the table allows selection of multiple rows,

otherwise the table allows selection of only one row at a time. When multiple selections are allowed, use CTRL-Click (press the CTRL key while clicking the left mouse button) on rows to select multiple rows.

See [TableClass Methods](#) for more information.

<- marker_width@

Returns the table row marker width

Class TableClass get

Format pixels = this.marker_width@

Description The get method marker_width@ returns the table row marker width in pixels. The table row markers appear on the left side of the table.

See [TableClass Methods](#) for more information.

<- model_is_data_request@

Returns the data request state

Class TableClass get

Format flag = this.model_is_data_request@

Description The get method model_is_data_request@ returns the data request state as a Boolean value. The method method returns TRUE if the table is in a data request state. The method returns FALSE if the table is not in a data request state. If the table is in a data request state the data_request_event is called on a vertical scroll action. Use the data_request_event to set the information displayed by the table after a scroll.

See [TableClass Methods](#) for more information.

<- rows_data@

Returns row information

Class TableClass get

Format flag = this.rows_data@(start, count)

Arguments start The starting row number. Rows are 0 based.
 count The quantity of rows from the starting row.

Description The get method `rows_data@` returns row information as a two-dimensional array. The first array dimension is the row, the second array dimension is the column information in each row. Use the method with the set method `one_row@` to verify and change row data.

See [TableClass Methods](#) for more information.

<- row_markers_is_suppressed@

Returns the row marker status

Class TableClass get

Format flag = this.row_markers_is_suppressed@

Description The get method `row_markers_is_suppressed@` returns the table row marker status as a Boolean value. The method returns TRUE if the display of row markers is suppressed. The method returns FALSE if the row markers are displayed in the table.

See [TableClass Methods](#) for more information.

<- row_numbers_is_enabled@

Returns the row number status

Class TableClass get

Format flag = this.row_numbers_is_enabled@

Description The get method `row_numbers_is_enabled@` returns the table row status as a Boolean value. The method returns TRUE if the display of row numbers is enabled. The method returns FALSE if the row numbers are not displayed in the table.

See [TableClass Methods](#) for more information.

<- rt_field_list@

Returns the associated Real Time fields

Class TableClass get

Format fieldArray = this.rt_field_list@

Description The get method `rt_field_list@` returns the Real Time fields associated with the table. The fields define the table columns for Real Time informations

See [TableClass Methods](#) for more information.

<- rt_record_list@

Returns the associated Real Time records

Class TableClass get

Format recArray = this.rt_record_list@

Description The get method `rt_record_list@` returns the Real Time records associated with the table. The fields define the table rows for Real Time informations

See [TableClass Methods](#) for more information.

<- selections@

Returns selected row numbers

Class TableClass get

Format selectArray = this.selections@

Description The get method `selections@` returns the selected row numbers as an array of numbers. Use the method with the set method `selections@` to verify and change the selected rows.

See [TableClass Methods](#) for more information.

<- table_data@

Returns table information

Class TableClass get

Format tableArray = this.table_data@

Description The get method table_data@ returns the table information as a two-dimensional array. The first array dimension is the row, the second array dimension is the column information in each row. Use the method with the set method table_data@ to verify and change all row data in the table.

See [TableClass Methods](#) for more information.

<- text_color@

Returns text color settings

Class TableClass get

Format colorArray = this.text_color@

Description The get method text_color@ returns a two dimensional array of text color settings. Text appears in the table.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [TableClass Methods](#) for more information.

`<- text_font_attrs@`

Returns text font information

Class TableClass get

Format format font_attrs_info@ fonts = this.text_font_attrs@

Description The get method `text_font_attrs@` returns all the text font information. Text appears in the table. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder.am` file. Include this file in any sources using the format. The `font_attrs_info@t` format is:

font_name The font name.

font_size The font point size.

bold The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.

italic The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.

shadow Not used.

See [TableClass Methods](#) for more information.

<- text_font_bold@

Returns text bold state

Class TableClass get

Format flag = this.text_font_bold@

Description The get method text_font_bold@ returns a Boolean value indicating the bold state of the object text. The method returns TRUE if the object text is bold, otherwise it returns FALSE. You can use this method with the set method text_font_bold@ to verify and change the object text bold state.

Text appears in the table.

For example, to make the object text bold use the following:

```
var flag
flag = this.text_font_bold@           'get method
IF NOT flag
    this.text_font_bold@ = TRUE       'set method
```

See [TableClass Methods](#) for more information.

<- text_font_italic@

Returns text italic state

Class TableClass get

Format flag = this.text_font_italic@

Description The get method text_font_italic@ returns a Boolean value indicating the italic state of the object text. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method text_font_italic@ to verify and change the object text italic state.

Text appears in the table.

For example, to make the object text italic use the following:

```
var flag
flag = this.text_font_italic@         'get method
IF NOT flag
```

```
    this.text_font_italic@ = TRUE           'set method
```

See [TableClass Methods](#) for more information.

<- text_font_name@

Returns text font name

Class TableClass get

Format name = this.text_font_name@

Description The get method `text_font_name@` returns the object text font name. You can use this method with the set method `text_font_name@` to verify and change the object text font.

Text appears in the table.

For example, to set the object text font to Courier use the following:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier"   'set method
```

See [TableClass Methods](#) for more information.

<- text_font_size@

Returns text font size

Class TableClass get

Format size = this.text_font_size@

Description The get method `text_font_size@` returns the object text font size. You can use this method with the set method `text_font_size@` to verify and change the object text size.

Text appears in the table.

For example, to set the object text size to 12 use the following:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
```

this.text_font_size@ = 12 'set method

See [TableClass Methods](#) for more information.

<- text_is_shadowed@

Returns text shadow state

Class TableClass get

Format flag = this.text_is_shadowed@

Description The get method text_is_shadowed@ returns a Boolean value indicating the shadow state of the object text. The method returns TRUE if the object text is shadowed, otherwise it returns FALSE. Use this method with the set method text_is_shadowed@ to verify and change the object text shadow state.

Text appears in the table. Text only appears shadowed on color displays.

See [TableClass Methods](#) for more information.

<- thickness@

Returns table thickness

Class TableClass get

Format pixels = this.thickness@

Description The get method height@ returns the table thickness in pixels. The thickness is the bevel appearance of the table sides.

See [TableClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class TableClass get

Format colorArray = this.title_color@

Description The get method title_color@ returns a two dimensional array of title color settings.

Title is the text that appears in the table heading.

The color settings are returned in the following format:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [TableClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class TableClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method `title_font_attrs@` returns all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format. The `font_attrs_info@t` format is:

<code>font_name</code>	The font name.
<code>font_size</code>	The font point size.
<code>bold</code>	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
<code>italic</code>	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
<code>shadow</code>	Not used.

See [TableClass Methods](#) for more information.

`<- title_font_bold@`

Returns title bold state

Class TableClass get

Format `flag = this.title_font_bold@`

Description The get method `title_font_bold@` returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method `title_font_bold@` to verify and change the object title bold state.

Title is the text that appears in the table heading.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [TableClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class TableClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears in the table heading.

For example, to make the object title italic use the following:

```
var flag
flag = this.title_font_italic@           'get method
IF NOT flag
    this.title_font_italic@ = TRUE       'set method
```

See [TableClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class TableClass get

Format name = this.title_font_name@

Description The get method title_font_name@ returns the object title font name. You can use this method with the set method title_font_name@ to verify and change the object text font.

Title is the text that appears in the table heading.

For example, to set the object title font to Courier use the following:

```
var name
name = this.title_font_name@           'get method
IF name <> "Courier"
    this.title_font_name@ = "Courier"  'set method
```

See [TableClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class TableClass get

Format size = this.title_font_size@

Description The get method title_font_size@ returns the object title font size. You can use this method with the set method title_font_size@ to verify and change the object text size.

Title is the text that appears in the table heading.

For example, to set the object title size to 12 use the following:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [TableClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class TableClass get

Format flag = this.title_is_shadowed@

Description The get method title_is_shadowed@ returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. Use this method with the set method title_is_shadowed@ to verify and change the object text shadow state.

Title is the text that appears in the table heading. Text only appears shadowed on color displays.

See [TableClass Methods](#) for more information.

<- top_row@

Returns top row

Class TableClass get

Format row = this.top_row@

Description The get method top_row @returns the top row displayed in the table as a number. The row numbers are zero based. Use the method with the set method top_row@ to verify and change the top row in the table.

See [TableClass Methods](#) for more information.

<- value@

Returns table information

Class TableClass get

Format tableArray = this.value@

Description The get method value@ returns the table information as a two-dimensional array. The first array dimension is the row, the second array dimension is the column information in each row. Use the method with the set method value@ to verify and change all row data in the table.

See [TableClass Methods](#) for more information.

<- vertical_grid_is_hidden@

Returns the vertical grid line status

Class TableClass get

Format flag = this.vertical_grid_is_hidden@

Description The get method vertical_grid_is_hidden@ returns the vertical grid line status as a Boolean value. The method returns TRUE if the vertical grid lines are hidden. The method returns FALSE if the vertical grid lines are displayed.

See [TableClass Methods](#) for more information.

<- vscroll_is_enabled@

Returns vertical scroll bar status

Class TableClass get

Format flag = this.vscroll_is_enabled@

Description The get method `vscroll_is_enabled@` returns the vertical scroll bar status as a Boolean value. The method returns TRUE if the vertical scroll bar is enabled and visible. The method returns FALSE if the scroll bar is disabled and hidden.

See [TableClass Methods](#) for more information.

<- vscroll_length@

Returns vertical scroll bar length

Class TableClass get

Format pixels = this.vscroll_length@

Description The get method `vscroll_length@` returns the vertical scroll bar length, in pixels.

See [TableClass Methods](#) for more information.

<- vscroll_origin@

Returns vertical scroll bar origin

Class TableClass get

Format pixels = this.vscroll_origin@

Description The get method `vscroll_origin@` returns the vertical scroll bar origin, in pixels, from the top right corner of the table.

See [TableClass Methods](#) for more information.

<- width@

Returns table width

Class TableClass get

Format pixels = this.width@

Description The get method width@ returns the table width in pixels. You can use this method with the set method width@ to verify and change the object's width.

For example, to make the table width at least 250 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [TableClass Methods](#) for more information.

-> button3_menu_info@

Sets pop up menu info

Class TableClass set

Format this.button3_menu_info@(format arrayof rminfo@ info)

Arguments info An array of rminfo@ information. The rminfo@ format is defined in the dialog_.am file, located in the *install_dir/axdata/elf* directory. The rminfo@ format is defined as:

format rminfo@

name, The name displayed in the menu.

macro_name, The macro or method name. Macro names must begin with the @ character to indicate it is a macro, not a method.

args,	An argument string.
active	A Boolean value, TRUE means the menu option is enabled, FALSE means the menu option is grayed and disabled.

Description The set method `button3_menu_info@` sets the pop up menu information. A pop up menu appears with a right mouse button press. A pop up menu is a free-floating menu associated with a dialog box control. A pop up menu can be activated when the mouse pointer is in the control area.

See [TableClass Methods](#) for more information.

-> `cell_editing_is_allowed@`

Sets the table cell edit status

Class TableClass set

Format `this.cell_editing_is_allowed@(flag)`

Arguments flag A Boolean value. TRUE makes the table cells editable, FALSE makes the tables cells uneditable. The default status is FALSE.

Description The set method `cell_editing_is_allowed@` sets the table cell edit status. Use this method with the get method `cell_editing_is_allowed@` to verify and change the table cell edit status. When table cells are editable, you can click in a cell and type at the text cursor.

For example, to enable table cell editing use the following:

```
if this.cell_editing_is_allowed@ = FALSE        'get method
   this.cell_editing_is_allowed@ = TRUE        'set method
```

See [TableClass Methods](#) for more information.

-> `column_resizing_is_allowed@`

Sets the column resizing status

Class TableClass set

Format `flag = this.column_resizing_is_allowed@`

Arguments flag A Boolean value. TRUE allows table column resizing, FALSE does not allow changes to column size.

Description The set method `column_resizing_is_allowed@` sets the column resizing status. See [TableClass Methods](#) for more information.

-> `control_color@`

Sets control color

Class TableClass set

Format `this.control_color@(type, c1, c2, c3, c4)`

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values.
See [TableClass Methods](#) for more information.

-> `control_color_cmyk@`

Sets control CMYK color

Class TableClass set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values.
See [TableClass Methods](#) for more information.

-> `control_color_is_workspace@`

Sets color used by object

Class TableClass set

Format `this.control_color_is_workspace@(flag)`

Arguments

<code>flag</code>	Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.
-------------------	---

Description The set method `control_color_is_workspace@` sets the color used by the object.
See [TableClass Methods](#) for more information.

-> `control_color_name@`

Sets control name color

Class TableClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name.

See [TableClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class TableClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method control_color_rgb@ sets the control color with RGB values.

See [TableClass Methods](#) for more information.

-> corner_pixmaps@

Sets bitmaps for table corner

Class TableClass set

Format this.corner_pixmaps@(pix)

Arguments pix An array of bitmap names.

Description The set method corner_pixmaps@ sets the bitmaps displayed in the upper left corner of the table, where the row markers and column headings intersect. Set the [marker_width@](#) to make all the corner bitmaps visible.

See [TableClass Methods](#) for more information.

-> data_set_field_name_list@

Sets data set fields used in the table

Class TableClass set

Format this.data_set_field_name_list@(fieldArray)

Arguments fieldArray An array of data set field names.

Description The set method data_set_field_name_list@ sets the data set fields displayed in the table. The current heading information is replaced with the field names.

See [TableClass Methods](#) for more information.

-> display@

Displays table

Class TableClass set

Format this.display@

Description The set method display@ displays the table.

See [TableClass Methods](#) for more information.

-> goto_cell@

Places cursor in cell

Class TableClass set

Format this.goto_cell@(row, col, start, end)

Arguments

row	The row number of the cell. The row index is 0-based.
col	The column number in the row. The column index is 0-based.
start	The character position in the cell at which to place the cursor. The character position is 1-based.
end	Set this argument to the same value as the start argument.

Description The set method `goto_cell@` places the cursor in a table cell before the specified character position.

See [TableClass Methods](#) for more information.

-> heading_info@

Sets table heading information

Class TableClass set

Format `this.heading_info@(valArray)`

Arguments `valArray` A two-dimensional array of heading information. The array has the following format:
`valArray[x,0]` The column heading string.
`valArray[x,1]` The column width, in pixels.

Description The get method `heading_info@` sets the table heading information. Use this method with the set method `heading_info@` to verify and change the table heading information.

See [TableClass Methods](#) for more information.

-> height@

Sets table height

Class TableClass set

Format `this.height@(value)`

Arguments `value` The height, in pixels, of the table.

Description The set method `height@` sets the table's height. You can use this method with the get method `height@` to verify and change the object's height.

For example, to make the table height at least 100 pixels you would use the method as follows:

```
var pixels
pixels = this.height@           'get method
IF pixels < 100
```


Description The set method `hscroll_length@` sets the horizontal scroll bar length.

See [TableClass Methods](#) for more information.

-> `hscroll_origin@`

Sets horizontal scroll bar origin

Class TableClass set

Format `this.hscroll_origin@(pixels)`

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the table.

Description The set method `hscroll_origin@` sets the horizontal scroll bar origin.

See [TableClass Methods](#) for more information.

-> `insert_text@`

Inserts text in the current cell

Class TableClass set

Format `this.insert_text@(text)`

Arguments text A text string.

Description The set method `insert_text@` inserts a text string in a cell at the current cell position.

Set [is_editable@](#) to TRUE before inserting text into the table.

See [TableClass Methods](#) for more information.

-> `is_editable@`

Sets table edit status

Class TableClass set

Format `this.is_editable@(flag)`

Arguments flag A Boolean value. TRUE enables table editing, FALSE disables table editing.

Description The set method `is_editable@` sets the table edit status. Use this method with the get method `is_editable@` to verify and change the table edit status.

For example, to enable table editing use the following:

```
if this.is_editable@ = FALSE            'get method
   this.is_editable@ = TRUE 'set method
```

See [TableClass Methods](#) for more information.

->is_read_only@

Sets the table to Read-only mode

Class TableClass set

Format `this.is_read_only@(flag)`

Arguments flag A Boolean. If TRUE, the table is set to Read Only. If FALSE, the table can be edited.

Description If the flag is set to TRUE, this method sets the table to read only mode. If a table is read-only, you cannot edit any table cells or highlight a row.

-> is_multi_select@

Sets table multiple row selection status

Class TableClass set

Format `this.is_multi_select@(flag)`

Arguments flag A Boolean value. TRUE allows selection of multiple rows, FALSE allows selection of only one row at a time. The default setting is FALSE.

Description The set method `is_multi_select@` sets the table multiple row selection status. When multiple selections are allowed, use CTRL-Click (press the CTRL key while clicking the left mouse button) on rows to select multiple rows.

See [TableClass Methods](#) for more information.

-> marker_pixmap@

Sets the table marker bitmaps

Class TableClass set

Format this.marker_pixmap@(row, pix)

Arguments row The starting row number for marker bitmaps. Row numbers are 0-based.
pix A two-dimensional array of bitmap names. The first dimension determines the row, the second dimension is the bitmap names for the row marker.

Description The set method marker_pixmap@ sets the table marker bitmaps. The table markers must be displayed, the [row markers is suppressed@](#) method must be set to FALSE. The [row numbers is enabled@](#) method must be set to FALSE, marker bitmaps cannot be displayed if row numbers are displayed. Table markers are not displayed if this method is called in the object's initialize_event, call this method after the object is initialized. For example, to set bitmaps for the first two row markers:

```
var pix
this.row_markers_is_suppressed@ = FALSE
this.row_numbers_is_enabled@ = FALSE
this.marker_width@ = 75
/* Assign bitmaps with one array */
pix[0] = {"bitmap1","bitmap2"}
/* Assign bitmaps individually */
pix[1][0] = "bitmap3"
pix[1][1] = "bitmap4"
this.marker_pixmap@(0,pix)
```

See [TableClass Methods](#) for more information.

-> marker_strings@

Sets the table marker strings

Class TableClass set

Format this.marker_strings@(row, pix)

Arguments row The starting row number for marker strings. Row numbers are 0-based.
pix An array of strings.

Description The set method `marker_strings@` sets the table marker strings. The table markers must be displayed, the `row markers is suppressed@` method must be set to FALSE. The `row numbers is enabled@` method must be set to FALSE, marker strings cannot be displayed if row numbers are displayed. Table markers are not displayed if this method is called in the object's `initialize_event`, call this method after the object is initialized. For example, to set strings for the first two row markers:

```
var pix
this.row_markers_is_suppressed@ = FALSE
this.row_numbers_is_enabled@ = FALSE
this.marker_width@ = 75
/* Assign strings with one array */
this.marker_strings@(0, {"Row 0", "Row1"} )
```

See [TableClass Methods](#) for more information.

-> `marker_width@`

Sets the table marker width

Class TableClass set

Format `this.marker_width@(pixels)`

Arguments pixels The marker width, in pixels.

Description The set method `marker_width@` sets the table marker width. The table markers must also be displayed. The set method `row_markers_is_suppressed@` should be set to FALSE, its default setting.

See [TableClass Methods](#) for more information.

-> `model_is_data_request@`

Sets the data request state

Class TableClass set

Format `this.model_is_data_request@(flag)`

Arguments `flag` A Boolean value. TRUE places the table in a data request state. FALSE does not place the table in a data request state. The default value is FALSE.

Description The set method `model_is_data_request@` sets the data request state. If the table is in a data request state the `data_request_event` is called on a vertical scroll action. Use the `data_request_event` to set the information displayed by the table after a scroll.

See [TableClass Methods](#) for more information.

-> `new_data@`

Adds data rows to a table

Class `TableClass` set

Format `this.new_data@(startRow, data)`

Arguments `startRow` An integer. The first row where the data should be inserted into the table.
`data` A two-dimensional array of data.

Description Adds data to a table. The start row is the row at which you want to begin adding data. The data array is two dimensional array containing information to be displayed in the table.

The location of the new data in the table is calculated as follows:

- The row is $(startrow+x)+1$.
- The column is y .

The following code puts the word "hello" at location (8,1) in the table, and the word "world" at location (8,2) in the table.

```
set clicked_event
```

```
var object tab  
var x
```

```
tab = this.sibling@("Table")
```

```
x[0,1] = "hello"  
x[0,2] = "world"
```

```
tab.new_data@(7, x)
endset
```

-> one_row@

Replaces row information

Class TableClass set

Format this.one_row(row, data)

Arguments row The row number. Rows are zero based.
 data The array of row information.

Description The get method one_row replaces information in a particular row.
See [TableClass Methods](#) for more information.

-> row_markers_is_suppressed@

Sets the row marker display status

Class TableClass set

Format this.row_markers_is_suppressed@(flag)

Arguments flag A Boolean value. TRUE suppresses the display of row markers. FALSE displays the row markers in the table. The default value is FALSE.

Description The set method row_markers_is_suppressed@ sets the table row marker display status. The set method marker_width@ must be greater than 0 for the row markers to appear in the table.

See [TableClass Methods](#) for more information.

-> row_numbers_is_enabled@

Sets the row number status

Class TableClass set

Format this.row_numbers_is_enabled@(flag)

Arguments flag A Boolean value. TRUE displays row numbers in the table. FALSE does not display row numbers in the table. The default value is FALSE.

Description The set method row_numbers_is_enabled@ sets the table row number status.

See [TableClass Methods](#) for more information.

-> rt_field_list@

Sets the associated Real Time fields

Class TableClass set

Format this.rt_field_list@(fieldArray)

Arguments fieldArray An array of Real Time field names.

Description The set method rt_field_list@ sets the Real Time fields associated with the table. The fields define the table columns for Real Time informations

See [TableClass Methods](#) for more information.

-> rt_record_list@

Sets the associated Real Time records

Class TableClass set

Format this.rt_record_list@(recArray)

Arguments recArray An array of Real Time record names.

Description The set method rt_record_list@ sets the Real Time records associated with the table. The records define the table rows for Real Time informations

See [TableClass Methods](#) for more information.

-> selections@

Selects multiple rows

Class TableClass set

Format this.selections@(rowArray)

Arguments rowArray An array of row numbers. Rows are zero based.

Description The set method selections@ selects multiple rows in the table.

See [TableClass Methods](#) for more information.

-> table_data@

Sets table information

Class TableClass set

Format this.table_data@(tableArray)

Arguments tableArray A two-dimensional array. The first array dimension is the row, the second array dimension is the column information in each row.

Description The set method table_data@ sets the table information. Use the method with the get method table_data@ to verify and change all row data in the table. Set the table heading information with [heading info@](#) before setting the table information with table_data@.

See [TableClass Methods](#) for more information.

-> text_color@

Sets text color

Class TableClass set

Format this.text_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `text_color@` sets the text color with an array of values. Text appears in the table.

See [TableClass Methods](#) for more information.

-> `text_color_cmyk@`

Sets text CMYK color

Class TableClass set

Format `this.text_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.

yellow The CMYK yellow value.
black The CMYK black value.

Description The set method `text_color_cmyk@` sets the text color with CMYK values. Text appears in the table.

See [TableClass Methods](#) for more information.

-> `text_color_name@`

Sets text name color

Class TableClass set

Format `this.text_color_name@(name)`

Arguments name The color name.

Description The set method `text_color_name@` sets the text color by name. Text appears in the table.

See [TableClass Methods](#) for more information.

-> `text_color_rgb@`

Sets text RGB color

Class TableClass set

Format `this.text_color_rgb@(red, green, blue)`

Arguments red The RGB red value.
green The RGB blue value.
blue The RGB green value.

Description The set method `text_color_rgb@` sets the text color with RGB values. Text appears in the table.

See [TableClass Methods](#) for more information.

-> text_font_attrs@

Sets text font information

Class TableClass set

Format this.text_font_attrs@(format font_attrs_info@ fonts)

Arguments

fonts	The font information:
font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method text_font_attrs@ sets all the text font information. Text appears in the table. The font_attrs_info@ format is defined in the */install_dir/axdata/elf/builder_.am* file. Include this file in any sources using the format.

See [TableClass Methods](#) for more information.

-> text_font_bold@

Sets text bold state

Class TableClass set

Format this.text_font_bold@(flag)

Arguments

flag	Indicates the bold state of the text. TRUE makes the text bold, FALSE unbolds the text.
------	---

Description The set method text_font_bold@ sets the text bold state. Text appears in the table. You can use this method with the get method text_font_bold@ to verify and change the object text bold state. Use the set method text_font_attrs@ to set all font information in one step.

See [TableClass Methods](#) for more information.

-> text_font_italic@

Sets text italic state

Class TableClass set

Format this.text_font_italic@(flag)

Arguments flag Indicates the italic state of the text. TRUE makes the text italic, FALSE makes the text standard.

Description The set method text_font_italic@ sets the italic state of the object text. Text appears in the table. You can use this method with the get method text_font_italic@ to verify and change the object text italic state. Use the set method text_font_attrs@ to set all font information in one step.

For example, to make the object text italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [TableClass Methods](#) for more information.

-> text_font_name@

Sets text font name

Class TableClass set

Format this.text_font_name@(name)

Arguments name A string for the font.

Description The set method text_font_name@ returns the object text font name. Text appears in the table. You can use this method with the get method text_font_name@ to verify and change the object text font. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text font to Courier you would use the method as follows:

```
var name
```

```
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [TableClass Methods](#) for more information.

-> text_font_size@

Sets text font size

Class TableClass set

Format this.text_font_size@(size)

Arguments size A numeric value for the font size.

Description The set method text_font_size@ sets the object text font size. Text appears in the table. You can use this method with the get method text_font_size@ to verify and change the object text size. Use the set method text_font_attrs@ to set all font information in one step.

For example, to set the object text size to you would use the method as follows:

```
var size
size = this.text_font_size@           'get method
IF size <> 12
    this.text_font_size@ = 12         'set method
```

See [TableClass Methods](#) for more information.

-> text_is_shadowed@

Sets text shadow state

Class TableClass set

Format this.text_is_shadowed@(flag)

Arguments flag Indicates the shadow state of the text. TRUE makes the text shadowed, FALSE makes the text standard.

Description The set method `text_is_shadowed@` sets the shadow state of the object text. Use this method with the get method `text_is_shadowed@` to verify and change the object text shadow state.

Text appears in the table. Text only appears shadowed on color displays.

See [TableClass Methods](#) for more information.

-> **thickness@**

Sets table thickness

Class TableClass set

Format `this.thickness@(pixels)`

Arguments pixels The thickness, in pixels.

Description The set method `height@` sets the table thickness in pixels. The thickness is the bevel appearance of the table sides.

See [TableClass Methods](#) for more information.

-> **title_color@**

Sets title color

Class TableClass set

Format `this.title_color@(type, c1, c2, c3, c4)`

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

c1 The RGB red value.

c2 The RGB blue value.

c3 The RGB green value.

The following values apply to CMYK color type:

c1 The CMYK cyan value.

c2 The CMYK magenta value.

c3 The CMYK yellow value.

c4 The CMYK black value.

The following values apply to HSB color type:

c1 The HSB hue value.

c2 The HSB saturation value.

c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values.

See [TableClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class TableClass set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

cyan	The CMYK cyan value.
magenta	The CMYK magenta value.
yellow	The CMYK yellow value.
black	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values.

See [TableClass Methods](#) for more information.

-> title_color_name@

Sets title name color

Class TableClass set

Format this.title_color_name@(name)

Arguments name The color name.

Description The set method title_color_name@ sets the title color by name.

See [TableClass Methods](#) for more information.

-> title_color_rgb@

Sets title RGB color

Class TableClass set

Format this.title_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [TableClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class TableClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments fonts The font information:
 font_name The font name.

font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	Not used.

Description The set method `title_font_attrs@` sets all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/axdata/elf/builder_.am` file. Include this file in any sources using the format.

See [TableClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class TableClass set

Format `this.title_font_bold@(flag)`

Arguments flag Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.

Description The set method `title_font_bold@` sets the title bold state. You can use this method with the get method `title_font_bold@` to verify and change the object title bold state. Use the set method `title_font_attrs@` to set all font information in one step.

See [TableClass Methods](#) for more information.

-> title_font_italic@

Sets title italic state

Class TableClass set

Format `this.title_font_italic@(flag)`

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method `title_font_italic@` sets the italic state of the object title. You can use this method with the get method `title_font_italic@` to verify and change the object title italic state. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [TableClass Methods](#) for more information.

-> `title_font_name@`

Sets title font name

Class TableClass set

Format `this.title_font_name@(name)`

Arguments name A string for the font.

Description The set method `title_font_name@` returns the object title font name. You can use this method with the get method `title_font_name@` to verify and change the object title font. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [TableClass Methods](#) for more information.

-> `title_font_size@`

Sets title font size

Class TableClass set

Format `this.title_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `title_font_size@` sets the object title font size. You can use this method with the get method `title_font_size@` to verify and change the object title size. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@            'get method
IF size <> 12
          this.title_font_size@ = 12            'set method
```

See [TableClass Methods](#) for more information.

-> `title_is_shadowed@`

Sets title shadow state

Class TableClass set

Format `this.title_is_shadowed@(flag)`

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method `title_is_shadowed@` sets the shadow state of the object title. Use this method with the get method `title_is_shadowed@` to verify and change the object title shadow state.

Title is the text that appears in the table heading. Text only appears shadowed on color displays.

See [TableClass Methods](#) for more information.

-> `top_row@`

Sets top row

Class TableClass set

Format `this.top_row@(row)`

Arguments row The row number. Rows are zero based.

Description The set method `top_row@` sets the top row displayed in the table. The row numbers are zero based. Use the method with the get method `top_row@` to verify and change the top row in the table.

See [TableClass Methods](#) for more information.

-> `value@`

Sets the table information if table data is NULL

Class `TableClass` set

Format `this.value@(tableArray)`

Arguments `tableArray` A two-dimensional array. The first array dimension is the row, the second array dimension is the column information in each row.

Description The set method `value@` sets the table information if the table currently contains NULL data. Use the [table_data@](#) method to change table data. Set the table heading information with [heading_info@](#) before setting the table information with `value@`.

See [TableClass Methods](#) for more information.

-> `vertical_grid_is_hidden@`

Sets the vertical grid line status

Class `TableClass` set

Format `this.vertical_grid_is_hidden@(flag)`

Arguments `flag` A Boolean value. TRUE hides the vertical grid lines, FALSE displays the vertical grid lines.

Description The set method `vertical_grid_is_hidden@` sets the vertical grid line status.

See [TableClass Methods](#) for more information.

-> vscroll_is_enabled@

Sets vertical scroll bar status

Class TableClass set

Format this.vscroll_is_enabled@(flag)

Arguments flag A Boolean value. TRUE makes the vertical scroll bar enabled and visible. FALSE makes the scroll bar disabled and hidden.

Description The set method vscroll_is_enabled@ sets the vertical scroll bar status.

See [TableClass Methods](#) for more information.

-> vscroll_length@

Sets vertical scroll bar length

Class TableClass set

Format this.vscroll_length@(pixels)

Arguments pixels The length of the scroll bar, in pixels

Description The set method vscroll_length@ sets the vertical scroll bar length.

See [TableClass Methods](#) for more information.

-> vscroll_origin@

Sets vertical scroll bar origin

Class TableClass set

Format this.vscroll_origin@(pixels)

Arguments pixels The origin of the scroll bar, in pixels, from the bottom left corner of the table.

Description The set method vscroll_origin@ sets the vertical scroll bar origin.

See [TableClass Methods](#) for more information.

-> width@

Sets table width

Class TableClass set

Format this.width@(value)

Arguments value The width, in pixels, of the table.

Description The set method width@ sets the tables's width. You can use this method with the get method width@ to verify and change the object's width.

For example, to make the table width at least 250 pixels you would use the method as follows:

```
var pixels
pixels = this.width@           'get method
IF pixels < 250
    this.width@ = 250         'set method
```

See [TableClass Methods](#) for more information.

<- button3_menu_state_event

Sets menu state before pop up menu is displayed

Class TableClass event

Format flag = this.button3_menu_state_event(function)

Arguments function The name of a pop up menu function.

Description The button3_menu_state_event is called by Applixware Builder before a pop up menu is displayed for a control. The event should be programmed to return either a Boolean value or NULL. The event should return TRUE if the menu item for the function should be active. The event should return FALSE if the menu item for the function should be grayed and inactive. The event should return NULL if the state of the menu item is unchanged.

For example, the following event sets the state of different menu items:

```
get button3_menu_state_event(function)
    case of function
    case "menu_1", "menu_2"
        return(NULL) ' No change in status
    case "menu_3"
        return(TRUE) ' Active menu item
    case "menu_4"
        return(FALSE) ' Inactive menu item
    endcase
endget
```

Use the [button3 menu info@](#) method to set the pop up menu information.

NOTE: The Debugger cannot access break points set in the `button3_menu_state_event`. Do not use break points in the `button3_menu_state_event` while you are debugging an application.

See [TableClass Methods](#) for more information.

<- error_event

Called for system errors

Class TableClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Appixware Builder.

Description The `error_event` is called by Appixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return TRUE if you handle the error in the error event. Have the method return FALSE if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [TableClass Methods](#) for more information.

-> **button_press_event**

Called when left mouse button pressed

Class TableClass event

Format this.button_press_event(value)

Arguments

value	A four element array with the following values:
value[0]	The row index, zero based.
value[1]	The column index, zero based.
value[2]	The start character number.
value[3]	The end character number.

Description The `button_press_event` is called by Applixware Builder when the left mouse button is pressed in a cell. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> **cell_changed_event**

Called when cell contents change

Class TableClass event

Format this.cell_changed_event(row, col, value)

Arguments

row	The cell row number. Row numbers are 0-based.
col	The cell column number. Column numbers are 0-based.
value	The new value in the cell.

Description The `cell_changed_event` is called by Applixware Builder when the contents of the current cell change. This event is called when you change focus to another cell. When you change the contents of a cell, then click in another table cell, the following sequence of events occurs.

1. A `cell_changed_event` is called for the current cell.
2. A [cell focus out event](#) is called for the current cell.

3. A **cell focus in event** is called for the new cell.

The **is_editable@** method must be set to TRUE for this event to be called. This is a user-defined event, place all actions you want performed in the event definition.

See **TableClass Methods** for more information.

-> cell_focus_in_event

Called when click in a table cell

Class TableClass event

Format this.cell_focus_in_event(row, col)

Arguments row The cell row number. Row numbers are 0-based.
col The cell column number. Column numbers are 0-based.

Description The cell_focus_in_event is called by Applixware Builder when you click in a table cell. The row and column of the cell are passed as arguments to the event. The **is_editable@** method must be set to TRUE for this event to be called. This is a user-defined event, place all actions you want performed in the event definition.

See **TableClass Methods** for more information.

-> cell_focus_out_event

Called when click in another table cell

Class TableClass event

Format this.cell_focus_out_event(row, col)

Arguments row The cell row number. Row numbers are 0-based.
col The cell column number. Column numbers are 0-based.

Description The cell_focus_out_event is called by Applixware Builder when you click in another table cell. The row and column of the current cell are passed to the event before focus changes to the new cell. This event is called, then a **cell focus in event** is called for the new table cell. The **is_editable@** method must be set to TRUE for this event to be called. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> column_resize_event

Called when column width changes

Class TableClass event

Format this.column_resize_event(value)

Arguments value A two element array with the following values:

value[0]	The column index, zero based.
value[1]	The column width, in pixels.

Description The column_resize_event is called by Applixware Builder when a column width in the table changes. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> double_click_event

Called when left mouse button double-clicked

Class TableClass event

Format this.double_click_event(value)

Arguments value Either a row index, zero based, or a two-element array containing the following:

value[0]	The row index, zero based.
value[1]	The column index, zero based.

Description The double_click_event is called by Applixware Builder when the left mouse button is double-clicked. The button_press_event is called first, then the double_click event. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class TableClass event

Format this.initialize_event

Description The initialize_event is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the table dimensions, position, and so on.

See [TableClass Methods](#) for more information.

-> request_data_event

Called to update data

Class TableClass event

Format this.request_data_event(value)

Arguments

value	A two element array with the following values:
value[0]	The starting row index, zero based.
value[1]	The number of rows

Description The request_data_event is called by Applixware Builder when a vertical table scroll occurs and the model_is_data_request@ is set to TRUE. This is a user-defined event, place all actions you want performed in the event definition. Use the event to place data in the rows exposed in the table viewing area.

See [TableClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class TableClass event

Format this.resize_event(width, height, old_width, old_height)

Arguments width The changed dialog box width.
 height The changed dialog box height.
 old_width The original dialog box width.
 old_height The original dialog box height.

Description The `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [TableClass Methods](#) for more information.

-> selection_changed_event

Called when table selection changes

Class TableClass event

Format `this.selection_changed_event(rowArray)`

Arguments rowArray An array of row numbers. Rows are zero based.

Description The `selection_changed_event` is called by Applixware Builder when the selections in the table change. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class TableClass event

Format `this.terminate_event`

Description The `terminate_event` is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class TableClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [TableClass Methods](#) for more information.

-> typing_event

Called when character typed in table cell

Class TableClass event

Format this.typing_event(row, col)

Arguments row The row index, zero based.
col The column index, zero based.

Description The typing_event is called by Applixware Builder when a character is typed in a table cell. The [is_editable@](#) method must be set to TRUE for this event to be called. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class TableClass event

Format this.update_event

Description The update_event is called by Applixware Builder to update a dialog box control. The [is_editable@](#) method must be set to TRUE for this event to be called. This is a user-defined event, place all actions you want performed in the event definition.

See [TableClass Methods](#) for more information.

<- control_color@

Returns color used by object

Class ToggleButtonClass get

Format colorArray = this.control_color@

Description The get method control_color@ returns the color used by the toggle button. The array information is returned as follows:

colorArray[0] The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

colorArray[1] The RGB red value.

colorArray[2] The RGB blue value.

colorArray[3] The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.

colorArray[2] The CMYK magenta value.

colorArray[3] The CMYK yellow value.

colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.

colorArray[2] The HSB saturation value.

colorArray[3] The HSB brightness value.

For a named color type, colorArray[1] is a string for the color name.

See [ToggleButtonClass Methods](#) for more information.

<- is_drop_shadowed@

Returns toggle button shadowed state

Class ToggleButtonClass get

Format flag = this.is_drop_shadowed@

Description The get method is_drop_shadowed@ returns a Boolean value indicating the toggle button shadowed state. If the method returns TRUE the toggle button is shadowed, otherwise the method returns FALSE. A drop shadow gives a toggle button a 3-dimensional appearance.

See [ToggleButtonClass Methods](#) for more information.

<- is_on@

Returns toggle button state

Class ToggleButtonClass get

Format flag = this.is_on@

Description The get method is_on@ returns a Boolean value indicating the button default state. If the method returns TRUE the toggle button is on, otherwise the method returns FALSE.

See [ToggleButtonClass Methods](#) for more information.

<- is_three_state@

Returns toggle button three-state status

Class ToggleButtonClass get

Format flag = this.is_three_state@

Description The get method `is_on@` returns a Boolean value indicating the button three-state status. If the method returns TRUE the toggle button is a three-state toggle button that can be set on, off, or grayed. If the method returns FALSE the toggle button is a standard toggle button that can only be set on or off.

See [ToggleButtonClass Methods](#) for more information.

<- title_color@

Returns title color settings

Class ToggleButtonClass get

Format colors = this.title_color@

Description The get method `title_color@` returns a two dimensional array of title color settings.

Title is the text that appears adjacent to the toggle button.

`colorArray[0]` The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

`colorArray[1]` The RGB red value.

`colorArray[2]` The RGB blue value.

`colorArray[3]` The RGB green value.

The following values apply to CMYK color type:

colorArray[1] The CMYK cyan value.
colorArray[2] The CMYK magenta value.
colorArray[3] The CMYK yellow value.
colorArray[4] The CMYK black value.

The following values apply to HSB color type:

colorArray[1] The HSB hue value.
colorArray[2] The HSB saturation value.
colorArray[3] The HSB brightness value.

If the color type is a named color, then the color name is returned as colorArray[1]. If the color type is a widget or work area color, the color type is the only value returned by the method.

See [ToggleButtonClass Methods](#) for more information.

<- title_font_attrs@

Returns title font information

Class ToggleButtonClass get

Format format font_attrs_info@ fonts = this.title_font_attrs@

Description The get method title_font_attrs@ returns all the title font information. The font_attrs_info@ format is defined in the */install_dir/daxdata/elf/builder_.am* file. Include this file in any sources using the format. The font_attrs_info@t format is:

font_name	The font name.
font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	The font shadow state as a Boolean value, TRUE means the font is shadowed, otherwise it sets FALSE.

See [ToggleButtonClass Methods](#) for more information.

<- title_font_bold@

Returns title bold state

Class ToggleButtonClass get

Format flag = this.title_font_bold@

Description The get method title_font_bold@ returns a Boolean value indicating the bold state of the object title. The method returns TRUE if the object title is bold, otherwise it returns FALSE. You can use this method with the set method title_font_bold@ to verify and change the object title bold state.

Title is the text that appears adjacent to the toggle button.

For example, to make the object title bold use the following:

```
var flag
flag = this.title_font_bold@           'get method
IF NOT flag
    this.title_font_bold@ = TRUE       'set method
```

See [ToggleButtonClass Methods](#) for more information.

<- title_font_italic@

Returns title italic state

Class ToggleButtonClass get

Format flag = this.title_font_italic@

Description The get method title_font_italic@ returns a Boolean value indicating the italic state of the object title. The method returns TRUE if the object text is italicized, otherwise it returns FALSE. You can use this method with the set method title_font_italic@ to verify and change the object title italic state.

Title is the text that appears adjacent to the toggle button.

For example, to make the object title italic use the following:

```
var flag
flag = this.title_font_italic@         'get method
IF NOT flag
```

```
        this.title_font_italic@ = TRUE           'set method
```

See [ToggleButtonClass Methods](#) for more information.

<- title_font_name@

Returns title font name

Class ToggleButtonClass get

Format name = this.title_font_name@

Description The get method `title_font_name@` returns the object title font name. You can use this method with the set method `title_font_name@` to verify and change the object text font.

Title is the text that appears adjacent to the toggle button.

For example, to set the object title font to Courier use the following:

```
var name
name = this.title_font_name@           'get method
IF name <> "Courier"
        this.title_font_name@ = "Courier"   'set method
```

See [ToggleButtonClass Methods](#) for more information.

<- title_font_shadow@

Returns title shadow type

Class ToggleButtonClass get

Format type = this.title_font_shadow@

Description The get method `title_font_shadow@` returns the title shadow type. An object title is shadowed when the [title is shadowed@](#) method is set to TRUE. The valid shadow types are:

Title is the text that appears adjacent to the toggle button.

See [ToggleButtonClass Methods](#) for more information.

<- title_font_size@

Returns title font size

Class ToggleButtonClass get

Format size = this.title_font_size@

Description The get method title_font_size@ returns the object title font size. You can use this method with the set method title_font_size@ to verify and change the object text size.

Title is the text that appears adjacent to the toggle button.

For example, to set the object title size to 12 use the following:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [ToggleButtonClass Methods](#) for more information.

<- title_is_shadowed@

Returns title shadow state

Class ToggleButtonClass get

Format flag = this.title_is_shadowed@

Description The get method title_is_shadowed@ returns a Boolean value indicating the shadow state of the object title. The method returns TRUE if the object title is shadowed, otherwise it returns FALSE. You can use this method with the set method title_is_shadowed@ to verify and change the object title shadow state.

Title is the text that appears adjacent to the toggle button.

See [ToggleButtonClass Methods](#) for more information.

<- value@

Returns toggle button value

Class ToggleButtonClass get

Format value = this.value@

Description The get method value@ returns the toggle button value. The possible toggle button values are:

0 Toggle button is off.

1 Toggle button is on.

2 Toggle button is grayed, available when [is three state@](#) is set to TRUE.

In the following example the toggle button title changes to reflect the toggle state.

```
var value
value = this.value@
IF value = 1
    this.title@ = "On"
ELSE
    this.title @= "Not On"
```

See [ToggleButtonClass Methods](#) for more information.

-> control_color@

Sets control color

Class ToggleButtonClass set

Format this.control_color@(type, c1, c2, c3, c4)

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color

- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `control_color@` sets the control color with an array of values. See [ToggleButtonClass Methods](#) for more information.

-> `control_color_cmyk@`

Sets control CMYK color

Class `ToggleButtonClass` set

Format `this.control_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `control_color_cmyk@` sets the control color with CMYK values. See [ToggleButtonClass Methods](#) for more information.

-> control_color_is_workspace@

Sets color used by object

Class ToggleButtonClass set

Format this.control_color_is_workspace@(flag)

Arguments flag Indicates the color used by the object. TRUE uses the work area color, FALSE uses the default widget color.

Description The set method control_color_is_workspace@ sets the color used by the object. See [ToggleButtonClass Methods](#) for more information.

-> control_color_name@

Sets control name color

Class ToggleButtonClass set

Format this.control_color_name@(name)

Arguments name The color name.

Description The set method control_color_name@ sets the control color by name. See [ToggleButtonClass Methods](#) for more information.

-> control_color_rgb@

Sets control RGB color

Class ToggleButtonClass set

Format this.control_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method `control_color_rgb@` sets the control color with RGB values.

See [ToggleButtonClass Methods](#) for more information.

-> `is_drop_shadowed@`

Sets toggle button shadowed state

Class ToggleButtonClass set

Format `this.is_drop_shadowed@(flag)`

Arguments `flag` A Boolean value. TRUE makes the toggle button the drop shadowed.

Description The set method `is_drop_shadowed@` sets the toggle button shadowed state. A drop shadow gives a toggle button a 3-dimensional appearance.

See [ToggleButtonClass Methods](#) for more information.

-> `is_on@`

Sets toggle button state

Class ToggleButtonClass set

Format `this.is_on@(flag)`

Arguments `flag` A Boolean value. TRUE sets the toggle button on, FALSE sets the toggle button off.

Description The set method `is_on@` sets the button state. If the method returns

See [ToggleButtonClass Methods](#) for more information.

-> `is_three_state@`

Sets toggle button three-state status

Class ToggleButtonClass set

Format `this.is_three_state@(flag)`

Arguments flag A Boolean value. TRUE sets the toggle button to a three-state toggle, while FALSE sets the toggle button to a standard two-state toggle.

Description The set method `is_on@` sets the toggle button three-state status. A three-state toggle button can be set on, off, or grayed. A standard two-state toggle button that can only be set on or off.

See [ToggleButtonClass Methods](#) for more information.

-> mux_toggle@

Makes toggle button diamond-shaped

Class ToggleButtonClass set

Format `this.mux_toggle@(flag)`

Arguments flag A Boolean value. TRUE makes the toggle button diamond-shaped, FALSE makes the toggle button square. The toggle button is square by default.

Description The set method `mux_toggle@` makes the toggle button diamond-shaped.

See [ToggleButtonClass Methods](#) for more information.

-> title_color@

Sets title color

Class ToggleButtonClass set

Format `this.title_color@(type, c1, c2, c3, c4)`

Arguments type The color type. The valid color types are:

- 1 RGB
- 2 Named color
- 3 Widget color
- 4 Work area color
- 5 HSB
- 6 CMYK

The following values apply to RGB color type:

- c1 The RGB red value.
- c2 The RGB blue value.
- c3 The RGB green value.

The following values apply to CMYK color type:

- c1 The CMYK cyan value.
- c2 The CMYK magenta value.
- c3 The CMYK yellow value.
- c4 The CMYK black value.

The following values apply to HSB color type:

- c1 The HSB hue value.
- c2 The HSB saturation value.
- c3 The HSB brightness value.

For a named color type, c1 is a string for the color name.

Description The set method `title_color@` sets the title color with an array of values.

See [ToggleButtonClass Methods](#) for more information.

-> `title_color_cmyk@`

Sets title CMYK color

Class `ToggleButtonClass` set

Format `this.title_color_cmyk@(cyan, magenta, yellow, black)`

Arguments

<code>cyan</code>	The CMYK cyan value.
<code>magenta</code>	The CMYK magenta value.
<code>yellow</code>	The CMYK yellow value.
<code>black</code>	The CMYK black value.

Description The set method `title_color_cmyk@` sets the title color with CMYK values.

See [ToggleButtonClass Methods](#) for more information.

-> title_color_name@

Sets title name color

Class ToggleButtonClass set

Format this.title_color_name@(name)

Arguments name The color name.

Description The set method title_color_name@ sets the title color by name.

See [ToggleButtonClass Methods](#) for more information.

-> title_color_rgb@

Sets control RGB color

Class ToggleButtonClass set

Format this.title_color_rgb@(red, green, blue)

Arguments red The RGB red value.
 green The RGB blue value.
 blue The RGB green value.

Description The set method title_color_rgb@ sets the title color with RGB values.

See [ToggleButtonClass Methods](#) for more information.

-> title_font_attrs@

Sets title font information

Class ToggleButtonClass set

Format this.title_font_attrs@(format font_attrs_info@ fonts)

Arguments fonts The font information:
 font_name The font name.

font_size	The font point size.
bold	The font bold state as a Boolean value, TRUE means the font is bold, otherwise it sets FALSE.
italic	The font italic state as a Boolean value, TRUE means the font is italic, otherwise it sets FALSE.
shadow	The font shadow state as a Boolean value, TRUE means the font is shadowed, otherwise it sets FALSE.

Description The set method `title_font_attrs@` sets all the title font information. The `font_attrs_info@` format is defined in the `/install_dir/daxdata/elf/builder_.am` file. Include this file in any sources using the format.

See [ToggleButtonClass Methods](#) for more information.

-> title_font_bold@

Sets title bold state

Class ToggleButtonClass set

Format `this.title_font_bold@(flag)`

Arguments flag Indicates the bold state of the title. TRUE makes the title bold, FALSE unbolds the title.

Description The set method `title_font_bold@` sets the title bold state. You can use this method with the get method `title_font_bold@` to verify and change the object title bold state. Use the set method `title_font_attrs@` to set all font information in one step.

See [ToggleButtonClass Methods](#) for more information.

-> title_font_italic@

Sets title italic state

Class ToggleButtonClass set

Format `this.title_font_italic@(flag)`

Arguments flag Indicates the italic state of the title. TRUE makes the title italic, FALSE makes the title standard.

Description The set method `title_font_italic@` sets the italic state of the object title. You can use this method with the get method `title_font_italic@` to verify and change the object title italic state. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to make the object title italic you would use the method as follows:

```
var flag
flag = this.text_font_italic@           'get method
IF NOT flag
    this.text_font_italic@ = TRUE       'set method
```

See [ToggleButtonClass Methods](#) for more information.

-> title_font_name@

Sets title font name

Class ToggleButtonClass set

Format `this.title_font_name@(name)`

Arguments name A string for the font.

Description The set method `title_font_name@` returns the object title font name. You can use this method with the get method `title_font_name@` to verify and change the object title font. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title font to Courier you would use the method as follows:

```
var name
name = this.text_font_name@           'get method
IF name <> "Courier"
    this.text_font_name@ = "Courier" 'set method
```

See [ToggleButtonClass Methods](#) for more information.

-> title_font_shadow@

Sets title shadow type

Class ToggleButtonClass set

Format `this.title_font_shadow@(type)`

Arguments type The shadow type. The defined shadow types are:

Description The set method `title_font_shadow@` sets the object title shadow type. An object title is shadowed when the [title is shadowed@](#) method is set to TRUE.

Title is the text that appears adjacent to the toggle button.

See [ToggleButtonClass Methods](#) for more information.

-> title_font_size@

Sets title font size

Class ToggleButtonClass set

Format `this.title_font_size@(size)`

Arguments size A numeric value for the font size.

Description The set method `title_font_size@` sets the object title font size. You can use this method with the get method `title_font_size@` to verify and change the object title size. Use the set method `title_font_attrs@` to set all font information in one step.

For example, to set the object title size to you would use the method as follows:

```
var size
size = this.title_font_size@           'get method
IF size <> 12
    this.title_font_size@ = 12         'set method
```

See [ToggleButtonClass Methods](#) for more information.

-> title_is_shadowed@

Sets title shadow state

Class ToggleButtonClass set

Format `this.title_is_shadowed@(flag)`

Arguments flag Indicates the shadow state of the title. TRUE makes the title shadowed, FALSE makes the title standard.

Description The set method `title_is_shadowed@` sets the shadow state of the object title. You can use this method with the get method `title_is_shadowed@` to verify and change the object title shadow state.

Title is the text that appears adjacent to the toggle button.

See [ToggleButtonClass Methods](#) for more information.

-> value@

Sets toggle button value

Class ToggleButtonClass set

Format `this.value@(value)`

Arguments value Sets the toggle button state. The toggle button state is one of the following values:

0 Toggle button set to off.

1 Toggle button set to on.

2 Toggle button is grayed, available when [is_three_state@](#) is set to TRUE.

Description The set method `value@` sets the toggle button value.

In the following example the toggle button state is set to the same value as its sibling toggle button if it is on.

```
var object sibling
var value
sibling = this.sibling@("siblingToggle")
IF sibling.value@ = 1
    this .value@ = 1
```

See [ToggleButtonClass Methods](#) for more information.

-> **changed_event**

Called when toggle button value changes

Class ToggleButtonClass event

Format this.changed_event(value)

Arguments value Indicates the toggle button state.

0	Toggle button is off.
1	Toggle button is on.
2	Toggle button is grayed, available when is three state@ is set to TRUE.

Description The changed_event is called by Applixware Builder when the the toggle button value changes. This is a user-defined event, place all actions you want performed in the event definition. The following is an example of a toggle button changed_event.

```
set changed_event(value)
  IF value = 1 ' Toggle is on
  {
    ' actions when toggle is on
  }
  ELSE '
  {
    ' actions when toggle is off or grayed
  }
```

See [ToggleButtonClass Methods](#) for more information.

<- **error_event**

Called for system errors

Class ToggleButtonClass event

Format flag = this.error_event(error_object)

Arguments error_object An object passed from Applixware Builder.

Description The `error_event` is called by Applixware Builder for posting object errors. If an `error_event` for the object does not exist, errors are passed to the default system error handler. This is a user-defined event, place all actions you want performed in the event definition. You can create an `error_event` for any object in your application.

The event should return a Boolean value. Have the event return `TRUE` if you handle the error in the error event. Have the method return `FALSE` if you want the default error handler.

Use [ErrorClass](#) methods to get error object information.

See [ToggleButtonClass Methods](#) for more information.

-> initialize_event

Called before displaying a control

Class `ToggleButtonClass` event

Format `this.initialize_event`

Description The `initialize_event` is called by Applixware Builder before displaying a control in a dialog box. This is a user-defined event, place all actions you want performed in the event definition, to initialize the toggle button color, position, and so on.

See [ToggleButtonClass Methods](#) for more information.

-> resize_event

Called when a dialog box is resized

Class `ToggleButtonClass` event

Format `this.resize_event(width, height, old_width, old_height)`

Arguments

<code>width</code>	The changed dialog box width.
<code>height</code>	The changed dialog box height.
<code>old_width</code>	The original dialog box width.
<code>old_height</code>	The original dialog box height.

Description The set event `resize_event` is called by Applixware Builder when a dialog box is resized. This is a user-defined event, place all actions you want performed in the event definition. Use the event to reposition widgets and to redraw the dialog box.

See [ToggleButtonClass Methods](#) for more information.

-> terminate_event

Called before closing or destroying dialog box

Class ToggleButtonClass event

Format this.terminate_event

Description The terminate_event is called by Applixware Builder before closing or destroying a dialog box. This is a user-defined event, place all actions you want performed in the event definition.

See [ToggleButtonClass Methods](#) for more information.

-> time_out_event

Called on object timer time-out

Class ToggleButtonClass event

Format this.time_out_event

Description The time_out_event is called by Applixware Builder on an object timer time-out. A time_out_event is generated when the object's timer expires. The time out interval is set with the [timer@](#) method. This is a user-defined event, place all actions you want performed in the event definition.

For example, a clock application would use this event to set the new time and display it. The timer@ method would be used to set the time interval to 1 second. A time_out_event would occur after 1 second.

See [ToggleButtonClass Methods](#) for more information.

-> update_event

Called to update dialog box control

Class ToggleButtonClass event

Format this.update_event

Description The `update_event` is called by Applixware Builder to update a dialog box control. This is a user-defined event, place all actions you want performed in the event definition.

See [ToggleButtonClass Methods](#) for more information.