Property Shorthand Notation in Window Event Code

The Event Handler compiler supports a shorthand syntax for setting and retrieving values of properties. The following examples summarize the shorthand syntax notation. The shorthand notation syntax is not case sensitive. Quotation marks are not used around property names.

The examples on this page apply only for BASIC+ code placed in window events, such as the CLICK event of a button. This notation will produce a compiler error for BASIC+ code not running in a window context, such as a stored procedure.

Symbols Used

The following information describes the symbols used in the shorthand notation:

Symbol	Symbol Name	Symbol Function
->	dash + greater than symbol	Get_Property(), Set_Property()
.	period	used current window name
@	"at" symbol	value of a variable name
@@	double "at" symbol	default property token

Examples of Syntax Usage

Retrieving a property (Get_Property()):

var = windowName.controlName->propertyName	;* Basic usage
var = .controlName->propertyName	;* Path prefix usage
var = @variable->@variable	;* Variable usage
var = @variable->propertyName	;* Variable usage
var = .controlName->@@	;* "Default Property" Token usage

Setting a property (Set_Property()):

windowName.controlName->propertyName = "value	;* Basic usage
.controlName->propertyName = "value"	;* Path prefix usage
@variable->@variable = "value"	;* Variable usage
@variable->propertyName = "value"	;* Variable usage
controlName->@@ = "value"	;* "Default Property" Token usage

Basic Usage

You can use the fully qualified control name and the actual property name as follows to extract the value of the property of a control in a window event handler.

Get_Property

SYNTAX:	var = windowName.controlName->propertyName
EXAMPLE:	state = CUST_ENTRY.STATE->text
EQUIVALENT:	state = Get_Property(@window:".STATE","TEXT")

This example extracts the value of the TEXT property from the STATE control in the CUST_ENTRY window, and places it in the "state" variable.

Set_Property

SYNTAX:	windowName.controlName->propertyName = var
EXAMPLE:	CUST_ENTRY.STATE->text = "NJ"
EQUIVALENT:	state = Set_Property(@window:".STATE","TEXT", "NJ")

This example sets the value of the TEXT property of the STATE control in the CUST_ENTRY window to "NJ".

PATH Prefix Usage

An alternative to using the fully qualified control name is to allow the event handler compiler to supply the name of the current window as the control name qualifier. The "." (period) character in the string enables this functionality. This technique shortens the length of the character string you specify in coding.

Get_Property

SYNTAX	var = .controlName->propertyName
EXAMPLE	state = .state->text
EQUIVALENT	state = Get_Property(@window:".STATE","TEXT")

This example extracts the value of the TEXT property from the STATE control in the current window, and places it in the "state" variable. Of course, a control with the name "STATE" should exist in the current window.

Set_Property

SYNTAX:	.controlName->propertyName = var
EXAMPLE:	.state->text = "NJ"
EQUIVALENT:	state = Set_Property(@window:".STATE","TEXT", "NJ")

This example sets the value of the TEXT property of the STATE control in the current window to "NJ". Of course, a control with the name "STATE" should exist in the current window.

Variable Usage

You can place a control name and/or property name into a variable and then use the variable in the shorthand notation. The "@" (at) character in the string enables this functionality.

Get_Property

SYNTAX:	var = @variable->@variable
	var = @variable->propertyName

EXAMPLE 1:	ctrlName = @window:".STATE"
	state = @ctrlName->text

EXAMPLE 2:	ctrlName = @window:".STATE"
	propName = "TEXT"
	state = @ctrlName->@propname

EQUIVALENT:	state = Get_Property(@window:".STATE","TEXT")

In both examples, the variable "state" receives the value of the TEXT property of the STATE control in the current window.

Set_Property

SYNTAX:	@variable->@variable = var @variable->propertyName = var
EXAMPLE 1:	ctrlName = @window:".STATE" @ctrlName->text = "NJ"

EXAMPLE 2:	ctrlName = @window:".STATE"
	
	propName = "TEXT"
	
	@ctrlName->@propname = "NJ"

EQUIVALENT:	state = Set_Property(@window:".STATE","TEXT", "NJ")

This example sets the value of the TEXT property of the STATE control in the current window to "NJ". Of course, a control with the name "STATE" should exist in the current window.

The @window variable may be referenced indirectly, by appending an @ sign, just like any other control name. The following code line, in a window event, reads the VISIBLE property of the window and assigns it to the variable winvisible.
For example, winvisible = @@window->visible

"Default Property" Token Usage

The default property token can be substituted for a variable representing the property name. The default property token is the "@" (at) character.

Get_Property

SYNTAX:	var = .controlName->@@
EXAMPLE:	state = .state->@@
EQUIVALENT:	state = Get_Property(@window:".STATE","DEFPROP")

The variable "state" receives the value of the default property of the STATE control in the current window.

Set_Property

SYNTAX:	.controlName->@@ = var
EXAMPLE:	.state->@@ = "NJ"
EQUIVALENT:	state = Set_Property(@window:".STATE","DEFPROP", "NJ")

This example sets the value of the DEFPROP property of the STATE control in the current window to "NJ". Of course, a control with the name "STATE" should exist in the current window.

A property in another window can be referenced without the @ prefix. Thus the following code gets the TEXT property of the window called APP BACKUP and assigns it to the variable appbackup_window_title.
For example, appbackup_window_title = APPBACKUP->Text

Properties in Expressions

Because of the way this syntax works for retrieving property values, it can be applied easily in logical expressions.   In the following example you can see the its application for setting a control property.

Example

```
// If  a the control in the current window has a null value in the TEXT property then set its TEXT property
value to "NJ"
if .STATE->TEXT = "" then
   .state->text = "NJ"
end
```