Programming Language
WinActor Scenario Script

NTT ADVANCED TECHNOLOGY CORPORATION

Trademarks

The names described below and other names of companies and products in this document are trademarks or registered trademarks of their respective companies. The ™, ®, and © marks are omitted in this document.

- WinActor is a registered trademark of NTT ADVANCED TECHNOLOGY CORPORATION.

- Microsoft, Windows*1, Internet Explorer, Excel, and VBScript*2 are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

  *1 The official name of Windows is Microsoft Windows Operating System.

  *2 The official name of VBScript is Microsoft Visual Basic Scripting Edition.

- The names of other companies and products are trademarks or registered trademarks of their respective companies.

About this document

This "Programming Language WinActor Scenario Script" (hereinafter referred to as "this manual") describes the programming language WinActor Scenario Script that can create scenarios of WinActor.

This manual is intended for advanced programmers who can create scenarios of WinActor programmatically rather than in a GUI.

Notes on this manual

- The copyright notice "Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION" attached to this manual and the provided software cannot be changed or deleted.
  The copyright of this manual belongs to NTT, Inc. and NTT ADVANCED TECHNOLOGY CORPORATION.
- The descriptions in this manual assume that users understand Windows operations and functions. For information that is not described in this manual, see the documents provided by Microsoft.

Contents

1. Introduction

The programming language WinActor Scenario Script (hereinafter referred to as WSS) is a procedural language for advanced programmers which is generated by converting a scenario of WinActor Ver.7.

WSS is written to a text file with the filename extension of .wss7, which is called a wss7 file.

A uss7 file is also generated together with a wss7 file. A set of a uss7 file and a wss7 file can be loaded into WinActor as a scenario.

For the procedure to save a scenario to uss7 and wss7 files or to load them into a scenario with WinActor, see "WinActor Operation Manual."

WSS is a procedural language similar to C programming language, but it has less flexibility because it is closely bound to the scenario of WinActor.

2. Terminology

The following terms are used in this manual. Details are described later.

Table 2-1. Terms

Term	Remarks
Flowchart	Points to a flowchart displayed in the flowchart area of WinActor
Identifier	
Identifier without a value	
Reserved identifier	
Action name	
Subroutine name	
JSON type name	Appears in JsonWrite adapter
Identifier with values	
Writable or read-only identifier	
Simple identifier	
Number identifier	
Special identifier	
Compiler generated working identifier	
Read-only identifier	
Predefined constant identifier	
Non-variable used in the form of assignment	Appears in preambles and const tuples
Attribute identifier	
String literal	
Simple string literal	
Verbatim string literal	
Block verbatim string literal	
Structure	
Preamble	
Adapter parameter list	
Verbatim tuple	
Const tuple	

3. Symbols and symbol strings

The symbols and symbol strings that have meaning in WSS are listed below.

Table 3-1. Symbols and symbol strings

Symbol & symbol string	Meaning	Usage
(	Left parenthesis	
)	Right parenthesis	
[	Left square bracket	
]	Right square bracket	
{	Left curly bracket	
}	Right curly bracket	
@(	At left parenthesis	Verbatim tuples
=	Assignment	
+	Addition	One of four arithmetic operators, unary operator, binary operator for constant expressions
-	Subtraction	One of four arithmetic operators, unary operator, binary operator for constant expressions
*	Multiplication	One of four arithmetic operators, binary operator for constant expressions
/	Division	One of four arithmetic operators, binary operator for constant expressions
+^	Addition (Integer)	One of four arithmetic operators, unary operator, binary operator for constant expressions
-^	Subtraction (Integer)	One of four arithmetic operators, unary operator, binary operator for constant expressions
*^	Multiplication (Integer)	One of four arithmetic operators, binary operator for constant expressions
/^	Division (Integer)	One of four arithmetic operators, binary operator for constant expressions
==	Equal to	Binary operator for constant/conditional expressions
!=	Not equal to	Binary operator for constant/conditional expressions
>=	Greater than or equal to	Binary operator for constant/conditional expressions

Symbol & symbol string	Meaning	Usage
>	Greater than	Binary operator for constant/conditional expressions
<	Less than	Binary operator for constant/conditional expressions
<=	Less than or equal to	Binary operator for constant/conditional expressions
!	Not	Unary operator for constant/conditional expression
&&	And	Binary operator for constant expressions
||	Or	Binary operator for constant expressions
~	Regular expression match	Binary operator for constant/conditional expressions
!~	Regular expression unmatch	Binary operator for constant/conditional expressions
'	Quote	Identifiers
_	Underscore	Identifiers
"	Double quote	Simple string literals
@"	Verbatim string literal start symbol	
@""""	Block verbatim string literal start symbol	
@""""""	Block verbatim string literal start symbol	
.	Period	
,	Comma	
;	Semicolon	
$	Dollar sign	Number identifiers, special identifiers, predefined constant identifiers
/*	Start of comment	
*/	End of comment	
//	Start of line comment	
\	Escape sign	See "Naming conventions for strings"
0 1 2 3 4 5 6 7 8 9	Number	

4. Data types

4.1 Comment

A part between /* and */ is skipped as a comment. It may span multiple lines.

A part from // to the end of the line is also skipped as a comment.

4.2 Number

Numbers include integers and floats, and are represented in decimal notation.

Table 4-1. Numerical compositions

Number	Composition
Integer	Digit sequence
Float	Digit sequence.Digit sequence
	Digit sequence e+/- Digit sequence
	Digit sequence.Digit sequence e+/- Digit sequence

"e" means "10 to the power of."

"e" may be a capital “E.”

"e" may be followed by "+" or "-."

"+" or "-" following "e" is optional.

The digit sequence on the right of "e" cannot be omitted.

The string literal of a number, which is embraced with half-width double quotes, may be used as a number in expressions including constant expressions.

Full-width digits, “+,” “-,” “E,” “e,” and “.” may also be used in notation of the number.

The simple string literal of a digit sequence with commas as thousand separators, which is embraced with half-width double quotes, is called a comma separated number. It may be used as a number in expressions including constant expressions.

Full-width digits and commas may also be used in notation of the comma separated numbers.

A comma separated number is always interpreted as a float number even if it includes no period, “e,” nor “E.” If it is used for an integer operand of an integer operation, a runtime error occurs.

4.3 Boolean

TRUE or FALSE. The strings TRUE and FALSE also work as Boolean.

4.4 Identifier

Identifiers are global. Variables inside a subroutine should also be declared in the variable part.

Reserved identifiers and predefined constant identifiers are case insensitive.

Other identifiers are case sensitive.

- Naming conventions for identifiers

Table 4-2. Naming conventions for identifiers

Naming convention	Description
[_alphanumeric]+	An identifier consisting of only alphanumeric characters and underscores. (Simple identifier)
'.*'	An identifier that encloses any characters with single quotes. Used when creating a name with Japanese characters. In particular, '' (consecutive two single quotes) can be used as an anonymous identifier to omit a value and indicate to use a default value.
$[_-alphanumeric.]+	An identifier that begins with $ and consists of alphanumeric characters, underscores, hyphens, and periods. Used for special identifiers and predefined constant identifiers.
$[numbers]+	Included above, but $[digit sequence] is called the number identifier. In the flowchart, it is treated as the name without $.
__work_[0-9][0-9][0-9][0-9]	A name with four digits appended to __work_. This identifier is generated by a compiler for its work. The behavior is not guaranteed if changed. Generated in the variable group __internal__.

In all naming conventions, the backslash character is used to escape the next character.

- Reserved identifier

The following identifiers are called reserved identifiers and are used in program syntax.
They are case insensitive.
The position where these can be used in the syntax is fixed.
It may be used as a simple identifier for a variable name in other places, but the behavior

when a reserved identifier is used as a subroutine name is not guaranteed.
Details of each reserved identifier are described later.

Table 4-3. Reserved identifiers

Name
var_group
const
window_rule
window_title
window_class
process_name
window_size
main
group
try
catch
callsub
return
call_scenario
scenario_return
dowhile
while
continue
break
count
start
end
counter
file
dbsource
user
password
table
template_and_data
json_object
switch
case
default

Name
if
then
else
winactor
sub
localvars
chkempty
floating
tag
rules
window_rule_ref
throw
error
subref
breakpoint_info
scenario_info
images
node_id_count
flow_divide_info
translation
true
false
istrue
isfalse
strcmp
strcasecmp

- Special identifier

Special identifiers can be used in expressions as special variables.
They should be written in all caps.

Some names include "-," and if a name matches a special identifier, "-" will be regarded as a part of the special identifier and will not be interpreted as a subtraction symbol.
For available special identifiers, see “Special variables” in the "WinActor Operation Manual."

Read only special variables cannot be assigned.

- Predefined constant identifier

The following identifiers are called predefined constant identifiers.
They are case insensitive.
These are prepared to describe numbers or strings by name, such as an action mode.
Predefined constant identifiers are all read-only.
They can be used in expressions and constant expressions.

Table 4-4. Predefined constant identifiers

Name	Value
$WIN32.WaitSettingOption	"specified_option_info"
$WIN32.WaitSettingScenario	"specified_scenario_info"
$WIN32.WaitSettingNode	"specified_node"
$SelectTabWin32.index	"index"
$SelectTabWin32.text	"text"
$GetListWin32.index	"index"
$GetListWin32.text	"text"
$SelectListWin32.index	"index"
$SelectListWin32.text	"text"
$GetListIE8.index	"index"
$GetListIE8.text	"text"
$SelectListIE8.index	"index"
$SelectListIE8.text	"text"
$TimerWait.Sleep	1
$TimerWait.Until	2
$TimerWait.Check	3
$TimerWait.ScenarioInfoDateFormat	"specified_scenario_info"
$TimerWait.OptionInfoDateFormat	"specified_option_info"
$TimerWait.OptionInfoTimeZone	"specified_option_info"
$TimerWait.DefaultTimeZone	"default_os"
$Window.Front	1
$Window.Behind	2
$Window.Enable	3
$Window.Disable	4
$Window.Appear	5
$Window.Disappear	6
$Window.WaitFor	1
$Window.CheckOnly	2
$ClipBoard.Set	1
$ClipBoard.Get	2

Name	Value
$IE.WaitSettingOption	"specified_option_info"
$IE.WaitSettingScenario	"specified_scenario_info"
$IE.WaitSettingNode	"specified_node"
$IEGetTableInfo.GetCell	"getcell"
$IEGetTableInfo.ExistCell	"existcell"
$IEGetTableInfo.GetRow	"getrow"
$IEGetTableInfo.GetColumn	"getcolumn"
$IEGetTableInfo.GetAll	"getall"
$WaitBox.Confirm	1
$WaitBox.Query	2
$GetDateTime.DateTime1	1
$GetDateTime.Date	2
$GetDateTime.Time	3
$GetDateTime.ScenarioInfoDateFormat	"specified_scenario_info"
$GetDateTime.OptionInfoDateFormat	"specified_option_info"
$GetDateTime.OptionInfoTimeZone	"specified_option_info"
$GetDateTime.DefaultTimeZone	"default_os"
$Calculate.Plus	1
$Calculate.Minus	2
$Calculate.Mul	3
$Calculate.Div	4
$Launcher.Single	1
$Launcher.Multi	2
$Launcher.WaitForEnd	3
$FullHalfWidth.ToFullWidth	"to_fullwidth"
$FullHalfWidth.ToHalfWidth	"to_halfwidth"
$Excel.GetValue	"get_value"
$Excel.SetValue	"set_value"
$Excel.RunMacro	"run_macro"
$ImageMatch.Check	1
$ImageMatch.LeftClick	2
$ImageMatch.RightClick	3
$ImageMatch.LeftDouble	4
$ImageMatch.RightDouble	5
$ImageMatch.Move	6
$ImageMatch.LeftTriple	7
$ImageMatch.RightTriple	8

Name	Value
$ImageMatch.LeftClickDrag	9
$ImageMatch.RightClickDrag	10
$ImageMatch.Same	0
$ImageMatch.Half	1
$ImageMatch.Quarter	2
$ImageMatch.StartPoint_LeftTop	"LeftTop"
$ImageMatch.StartPoint_LeftBottom	"LeftBottom"
$ImageMatch.StartPoint_RightTop	"RightTop"
$ImageMatch.StartPoint_RightBottom	"RightBottom"
$ImageMatch.Coordinate_Direct	"DIRECT"
$ImageMatch.Coordinate_Percent	"PERCENT"
$ImageMatch.Path_File	"FilePath"
$ImageMatch.Path_Folder	"FolderPath"
$ImageMatch.SelectShape_Ellipse	"Ellipse"
$ImageMatch.SelectShape_Rectangle	"Rect"
$OutlineMatch.Check	1
$OutlineMatch.LeftClick	2
$OutlineMatch.RightClick	3
$OutlineMatch.LeftDouble	4
$OutlineMatch.RightDouble	5
$OutlineMatch.Move	6
$OutlineMatch.LeftTriple	7
$OutlineMatch.RightTriple	8
$OutlineMatch.LeftClickDrag	9
$OutlineMatch.RightClickDrag	10
$OutlineMatch.Same	0
$OutlineMatch.Half	1
$OutlineMatch.Quarter	2
$OutlineMatch.LowPrecision	1
$OutlineMatch.MiddlePrecision	2
$OutlineMatch.HighPrecision	3
$OutlineMatch.StartPoint_LeftTop	"LeftTop"
$OutlineMatch.StartPoint_LeftBottom	"LeftBottom"
$OutlineMatch.StartPoint_RightTop	"RightTop"
$OutlineMatch.StartPoint_RightBottom	"RightBottom"
$OutlineMatch.Coordinate_Direct	"DIRECT"
$OutlineMatch.Coordinate_Percent	"PERCENT"

Name	Value
$OutlineMatch.Path_File	"FilePath"
$OutlineMatch.Path_Folder	"FolderPath"
$OCRMatch.Check	1
$OCRMatch.LeftClick	2
$OCRMatch.RightClick	3
$OCRMatch.LeftDouble	4
$OCRMatch.RightDouble	5
$OCRMatch.Move	6
$OCRMatch.LeftTriple	7
$OCRMatch.RightTriple	8
$OCRMatch.LeftClickDrag	9
$OCRMatch.RightClickDrag	10
$OCRMatch.StartPoint_LeftTop	"LeftTop"
$OCRMatch.StartPoint_LeftBottom	"LeftBottom"
$OCRMatch.StartPoint_RightTop	"RightTop"
$OCRMatch.StartPoint_RightBottom	"RightBottom"
$OCRMatch.Coordinate_Direct	"DIRECT"
$OCRMatch.Coordinate_Percent	"PERCENT"
$HTTP.Get	"get"
$HTTP.Put	"put"
$HTTP.Post	"post"
$HTTP2.RawFormat	false
$HTTP2.JSONFormat	true
$MailReceive.OneByOne	"ONE_BY_ON"
$MailReceive.GetAll	"GET_ALL"
$MailReceive.NumOnly	"NUM_ONLY"
$MailReceive.Wait	"RECEIVE_WAIT"
$MailReceive.Error	"RETURN_ERROR"
$MailReceive.MailNum	"RETURN_MAIL_NUM"
$MailSelect.Top	"MAIL_TOP"
$MailSelect.NoProcessedTop	"MAIL_NO_PROCESSED"
$MailSelect.ProcessedTop	"MAIL_PROCESSED"
$MailSelect.Next	"MAIL_NEXT"
$MailSelect.NextNoProcessed	"MAIL_NEXT_NO_PROCESSED"
$MailSelect.NextProcessed	"MAIL_NEXT_PROCESSED"
$MailStatusChg.Processed	"PROCESSED"
$MailStatusChg.NoProcessed	"NO_PROCESSED"

Name	Value
$MailCopyClip.UniqueID	"UID"
$MailCopyClip.FolderName	"DIR"
$MailCopyClip.Status	"STAT"
$MailCopyClip.SendDate	"SEND_DATE"
$MailCopyClip.From	"FROM"
$MailCopyClip.Subject	"SUBJECT"
$MailCopyClip.Body	"MESSAGE"
$MailCopyClip.NumberOfAttached	"ATTACHMENT"
$MailRule.Subject	"SUBJECT"
$MailRule.To	"TO"
$MailRule.From	"FROM"
$MailRule.Include	"CONTAIN"
$MailRule.AtFirst	"FIRST"
$MailRule.AtLast	"LAST"
$MailRule.Equal	"EQUAL"
$MailRule.Regex	"REGULAR_EXPRESSION"
$MailAuth.UserPass	"USER_PASS"
$MailRule.APOP	"APOP"
$GmailReceive.OneByOne	"ONE_BY_ON"
$GmailReceive.GetAll	"GET_ALL"
$GmailReceive.NumOnly	"NUM_ONLY"
$GmailReceive.Wait	"RECEIVE_WAIT"
$GmailReceive.Error	"RETURN_ERROR"
$GmailReceive.MailNum	"RETURN_MAIL_NUM"
$WindowRule.Unspecified	"NOSPECIFIED"
$WindowRule.ExactMatch	"STRING_EQUALS"
$WindowRule.PartialMatch	"CONTAINS"
$WindowRule.AtFirst	"BEGINS"
$WindowRule.AtLast	"ENDS"
$WindowRule.Regex	"REGEX"
$WindowRule.Equal	"EQUALS"
$WindowRule.GTE	"GTE"
$WindowRule.LTE	"LTE"
$SCENARIO_INFO.DefaultTimeZone	"default_os"
$SCENARIO_INFO.DefaultDateFormat	"WinActor.Main.Common.TimeFormat"
$SCENARIO_INFO.WaitSettingOption	"specified_option_info"
$SCENARIO_INFO.WaitSettingScenario	"specifed_scenario_info"

Name	Value
$UIA.CommonPattern	"CommonPattern"
$UIA.WaitSettingOption	"option"
$UIA.WaitSettingScenario	"scenario"
$UIA.WaitSettingNode	"node"
$UIA.WaitForWindow	"window"
$UIA.WaitForControl	"element"
$UIA.ExpandCollapsePattern	"ExpandCollapsePattern"
$UIA.InvokePattern	"InvokePattern"
$UIA.ScrollPattern	"ScrollPattern"
$UIA.SelectionPattern	"SelectionPattern"
$UIA.SelectionItemPattern	"SelectionItemPattern"
$UIA.TogglePattern	"TogglePattern"
$UIA.ValuePattern	"ValuePattern"
$UIA.MouseExtensionPattern	"MouseExtensionPattern”
$UIA.UnknownPattern	"Unknown"
$UIA.GetName	"GetName"
$UIA.Expand	"Expand"
$UIA.Collapse	"Collapse"
$UIA.Invoke	"Invoke"
$UIA.IsHorizontallyScrollable	"IsHorizontallyScrollable"
$UIA.GetHorizontalViewportRatio	"GetHorizontalViewportRatio"
$UIA.GetHorizontalViewportSize	"GetHorizontalViewportSize"
$UIA.HorizontalScroll	"HorizontalScroll"
$UIA.IsVerticallyScrollable	"IsVerticallyScrollable"
$UIA.GetVerticalViewportRatio	"GetVerticalViewportRatio"
$UIA.GetVerticalViewportSize	"GetVerticalViewportSize"
$UIA.VerticalScroll	"VerticalScroll"
$UIA.TwoWayScroll	"TwoWayScroll"
$UIA.IsMultiSelectable	"IsMultiSelectable"
$UIA.IsSelectionNeeded	"IsSelectionNeeded"
$UIA.GetSelectionByTexts	"GetSelectionByTexts"
$UIA.GetSelectionByIndexes	"GetSelectionByIndexes"
$UIA.SelectItemByText	"SelectItemByText"
$UIA.SelectItemByIndex	"SelectItemByIndex"
$UIA.IsSelected	"IsSelected"
$UIA.SelectAdditionally	"SelectAdditionally"
$UIA.Unselect	"Unselect"

Name	Value
$UIA.SelectOne	"SelectOne"
$UIA.Toggle	"Toggle"
$UIA.GetToggleState	"GetToggleState"
$UIA.IsReadOnly	"IsReadOnly"
$UIA.GetValue	"GetValue"
$UIA.SetValue	"SetValue"
$UIA.GetBoxCenterPosition	"GetBoxCenterPosition"
$UIA.GetBoxPositions	"GetBoxPositions"
$UIA.MoveToTheElementCenter	"MoveMousePosotionToCenter”
$UIA.LeftClickTheElementCenter	"MoveMousePosotionToCenterAndClick LeftButton"
$UIA.RightClickTheElementCenter	"MoveMousePosotionToCenterAndClick RightButton"
$UIA.Unknown	"Unknown"
$UIA.LargeDecrement	"LargeDecrement"
$UIA.SmallDecrement	"SmallDecrement"
$UIA.LargeIncrement	"LargeIncrement"
$UIA.SmallIncrement	"SmallIncrement"
$UIA.NoAmount	"NoAmount"
$UIA.ModeNormal	"SetValueModeNormal"
$UIA.ModeKeyEvent	"SetValueModeKeyEvent"
$UIA.ModeExcelCell	"SetValueModeExcelCell"
$UIADUMP.WaitSettingOption	"specified_option_info"
$UIADUMP.WaitSettingScenario	"specified_scenario_info"
$UIADUMP.WaitSettingNode	"specified_node"
$TAG.TopLeft	"AREA_TOP_LEFT"
$TAG.TopCenter	"AREA_TOP_CENTER"
$TAG.TopRight	"AREA_TOP_RIGHT"
$TAG.CenterLeft	"AREA_CENTER_LEFT"
$TAG.CenterCenter	"AREA_CENTER_CENTER"
$TAG.CenterRight	"AREA_CENTER_RIGHT"
$TAG.BottomLeft	"AREA_BOTTOM_LEFT"
$TAG.BottomCenter	"AREA_BOTTOM_CENTER"
$TAG.BottomRight	"AREA_BOTTOM_RIGHT"
$EVENT.UpdateFile	"UPDATE_FILE"
$EVENT.UpdateFolder	"UPDATE_FOLDER"
$EVENT.SpecifiedTime	"SPECIFIED_TIME"
$EVENT.Monthly	"MONTHLY"

Name	Value
$EVENT.Weekly	"WEEKLY"
$EVENT.Everyday	"EVERYDAY"
$EVENT.Hour	"HOUR"
$EVENT.Minute	"MINUTE"
$EVENT.WindowState	"WINDOW_STATE"
$EVENT.Mail	"MAIL"
$EVENT.DAY	"USER"
$EVENT.STARTDAY	"START"
$EVENT.LASTDAY	"END"
$EVENT.Mon	"1"
$EVENT.Tue	"2"
$EVENT.Wed	"4"
$EVENT.Thu	"8"
$EVENT.Fri	"16"
$EVENT.Sat	"32"
$EVENT.Sun	"64"
$SOCKET.ActionException	"アクション例外"
$SOCKET.ActionExceptionEN	"ActionException"
$SOCKET.CharASCII	"ASCII"
$SOCKET.CharUTF-16LE	"utf-16LEN"
$SOCKET.CharUTF-16LE-BOM	"utf-16LE "
$SOCKET.CharUTF-16BE	"utf-16BEN"
$SOCKET.CharUTF-16BE-BOM	"utf-16BE"
$SOCKET.CharUTF-8	"utf-8n"
$SOCKET.CharUTF-8-BOM	"utf-8"
$SOCKET.CharShift-JIS	"shift_jis"
$SOCKET.CharEUC-JP	"euc-jp"
$SOCKET.NewLineCRLF	"CRLF"
$SOCKET.NewLineCR	"CR"
$SOCKET.NewLineLF	"LF"
$SOCKET.NewLineNONE	"NONE"
$SOCKET.EndFin	"Fin"
$SOCKET.EndSize	"Size"
$SOCKET.EndEmpty	"Empty"

- Action name

The identifiers that represent action names are as follows. They are case insensitive.

Table 4-5. Action names

Name
ClickWin32
SetTextWin32
SelectListWin32
SelectTabWin32
EmulationWin32
GetTextWin32
GetListWin32
GetCheckWin32
GetEnableWin32
GetAllListWin32
ClickIE8
SetTextIE8
SelectListIE8
GetTextIE8
GetListIE8
GetCheckIE8
GetEnableIE8
GetTableinfoIE8
GetAllListIE8
SendText
ImageMatch
OutlineMatch
OCRMatch
WindowStateWait
TimerWait
InputBox
SelectBox
WaitBox
SetVariable
CopyVariable
GetDateTime
GetUserName
Calculate
CountUp
PlaySound

Name
Beep
Speaker
Launcher
Clipboard
SetToClipboard
GetFromClipboard
Script
TextConvert
Excel
MailReceive
MailSelect
MailStatusChg
MailSync
MailRemove
MailRemoveProcessed
MailCopyClip
MailAttachName
MailGetInfo
MailReceiveSet
MailReceiveImport
GmailReceiveSet
GmailReceive
GmailSendSet
GmailSend
Http
Http2
JsonWrite
JsonRead
UIAutomation
UiaDump
UiaExpandMenu
UiaCollapseMenu
UiaClick
UiaGetItemTextInList
UiaGetItemIndexInList
UiaGetAllItemTextInList
UiaSelectItemTextInList

Name
UiaSelectItemIndexInList
UiaSelectTab
UiaSelectRadioButton
UiaGetText
UiaSetText
UiaSetChecked
EventAdd
EventsWatch
EventRemove
EventsIgnore
Socket

4.5 String

- String literals

Table 4-6. String literals

Name
Simple string literal
Verbatim string literal
Block verbatim string literal

- Naming conventions for strings

Table 4-7. Naming conventions for strings

Naming convention	Description
".*"	Characters enclosed in double quotes make a simple string literal. It may span multiple lines. The \ (backslash) is an escape character and has the following meanings with the next character. For a string that span multiple lines, the escape character at the end of a line is ignored. Even if the escape character is written at the end of a line, it is not connected to the next line. \\ backslash \0 null \b backspace \r return \n linefeed

Naming convention	Description
	\f formfeed \t horizontal tab \v vertical tab
@".*"	A string starting with @" and ending with " is a verbatim string literal. A backslash character has no special meaning and is treated as a backslash character itself. To include a double quote in the string, write "" (consecutive two double quotes). Assumed to be used for a path name of a file or a folder.
@"""[\s]*$ … """	When a line ends with @""", all characters from the next line to the line of only """ are treated as a string. This is a block verbatim string literal. No escape sequence is available. Assumed to be used for script or annotation of a script adapter.
@"""""[\s]*$ … ^"""""[\s]*,?[\s]*$	When a line ends with @""", all characters from the next line to the line of only """ are treated as a string. This is a block verbatim string literal. No escape sequence is available. Assumed to be used for Python script or annotation of a script adapter. When either of these notations are included in an annotation or a comment of Python script, the result is indefinite.

4.6 Structure

4.6.1 Preamble

A preamble is used to describe node attributes such as "name" and "comment."

If the "isclosed" attribute is set to true, a node will be displayed in a closed style in the flowchart.

In particular, the ID attribute is referred to by a sticky note or a breakpoint.

The nodes referring to the ID attribute are described later.

A position where a preamble can be placed is fixed.

A preamble is optional.

If omitted, the node name in the flowchart will be a fixed name for each node, and the comment will be empty.

Details of the constant expression are described later.

- How to write a preamble

```
[ Attribute identifier = Constant expression, ... ]
```

Write the "Attribute identifier = Constant expression" parts separated by commas inside the left and right square brackets.

The number of "Attribute identifier = Constant expression" parts can be 0.

Valid attribute identifiers are defined depending on the place of the preamble.

Usually in a node, "name" and "comment" attributes are valid.

"name" attribute corresponds to the name in a node property and "comment" attribute corresponds to the comment.

The attributes specific to each preamble are described later.

- Example of a preamble

```
[ name = "Decision group", comment = "Decision for each input", isclosed = true ]
```

4.6.2 Adapter parameter list

- How to write an adapter parameter list

```
( Attribute identifier|String<Identifier> = Actual parameter , ... )
```

Start with (and end with). Write adapter parameters inside the parentheses separated by commas. Some are only ().

"<Identifier>" has meaning only in a specific adapter. (Described later.)

The "Attribute identifier|String<Identifier> = " part is optional depending on where it is used.

Write the following elements in the "Actual parameter" part. You cannot write a conditional expression.

Table 4-8. Elements of actual parameters

Element	Remarks
Expression	Expressions may not be allowed depending on the attribute of the adapter. (Described later)
Constant expression	
Verbatim tuple	
Adapter parameter list	

- Examples of adapter parameter lists

Example1: WinActor.SendText adapter

```
(
    window_rule_ref = "Untitled-Notepad",
    control = (instance<true> = 0, text<true> = "Untitled - Notepad", position<true> = ret01),
    value = var01,
    sendcr = true,
    verify = true,
    capture = (imageid = "img_20190613172532792", x = 409, y = 10)
      //  An example of using an adapter parameter list in the actual parameter part
)
```

Example 2: WinActor.EmulationWin32 adapter parameter list

```
(
    window_rule_ref = "Window",
    action = @(Wait, 300),        //  An example of using a verbatim tuple
    capture = (imageid = "_", x = 0, y = 0)
)
```

Example 3: WinActor.SelectBox adapter parameter list

```
(
    message = "Select",
    items = ("Red", "Blue", "White")   //  An example of omitting "attribute identifier ="
)
```

4.6.3 Verbatim tuple

A verbatim tuple is a structure that consists of an identifier, a string, or a number.

It may include a constant expression to get a string or a number.

It is assumed to be used for writing mouse actions of the emulation adapter.

- How to write a verbatim tuple

```
@( Identifier|String|Number|(Constant expression) , ... )
```

Start with @(and end with). Write identifiers, strings, or numbers inside the parentheses separated by commas.

Enclose constant expressions in (and). If not enclosed, a syntax error will occur.

An identifier is simply treated as a name. Even if the identifier has been declared as a constant, the value will not be used.

Details of the constant declaration are described later.

- Example of verbatim tuples

```
action = (@(Mouse, L, DOWN, 421, 28, LEFTTOP, D, D),
```

```
@(Mouse, L, UP, 421, 28, LEFTTOP, D, D),
@(Wait, 1719),
@(Mouse, L, DOWN, 62, 459, LEFTTOP, D, D),
@(Mouse, L, UP, 62, 459, LEFTTOP, D, D),
@(Wait, 695),
@(Mouse, L, DOWN, 309, 446, LEFTTOP, D, D),
@(Mouse, L, UP, 309, 446, LEFTTOP, D, D),
@(Wait, ('Short wait') ))
```

"Short wait" is assumed to be declared as a constant.

4.6.4 Const tuple

A const tuple is used in the following places. It gives a constant to a name.

Table 4-9. Where to use constant tuples

Where to use constant tuples
Window match rule part
Sticky note for floating part
Breakpoint information part
Scenario information part
Image part
Flowchart information part

- How to write a constant tuple

```
( Attribute identifier|String = Constant expression, ... )
```

Start with (and end with). Write constant expressions inside the parentheses separated by commas.

An attribute identifier or a string is written on the left side. If its string expression is the same, it is regarded as the same name. (Example: abc and "abc" are the same)

An error will occur if attribute identifiers or strings are duplicated.

5. Scenario

5.1 Scenario composition

A scenario is composed of multiple types of parts and should be written in the following order.

The required number of parts is determined by each type of the part.

Table 5-1. Scenario composition

No.	Name	Required number
1	Variable part	0 or more
2	Window match rule part	0 or more
3	Main part	1
4	Floating part	0 or more
5	Subroutine part	0 or more
6	WinWatcher part	0 or 1
7	Breakpoint information part	0 or 1
8	Scenario information part	1
9	Image part	0 or 1
10	Flowchart information part	1
11	Word dictionary part	0 or 1

5.2 Variable part

- Syntax

```
Var_group Group name = ( Variable declaration , ...  )
```

- Description

Write variable declarations and constant declarations.
You can write multiple variables in one group.

Write "Group name" with a string or an identifier.
"Group name =" is optional.
You can write 0 or more variable declarations by separating them with commas.

5.2.1 Variable declaration

- Syntax

```
const Identifier = Constant expression Preamble
```

- Description

Variable declaration and constant declaration are possible.

It becomes a variable declaration if "const" is omitted, and it becomes a constant declaration if "const" is given.

In the case of variable declaration, "= Constant expression" is optional. If it is written, it becomes the initial value.

The initial value is required for a constant declaration.

Another value cannot be assigned to an identifier with "const."

An identifier with "const" can be used in a constant expression.

The identifier must to be unique within a scenario. It cannot be the same even in different variable groups.

The initial value of a variable in which the mask column is checked in the Variable list pane of WinActor will not be output to the wss7 file.

"mask = true" will be output to the preamble.

If "mask = true" is specified in the preamble of the variable declaration and the initial value is omitted, WinActor will use the initial value stored in the uss7 file.

5.3 Window match rule part

- Syntax

```
Window_rule WinID name Preamble = (
    window_title = Const tuple,
    window_class = Const tuple,
    process_name = Const tuple,
    window_size  = Const tuple
)
```

- Description

The window match rule part is assumed to be generated by WinActor itself and edited in WSS as needed.

Write WinID name with a string. An error will occur if it is duplicated in the window match rule part.
The only valid attribute in the preamble is the comment.

5.3.1 window_title const tuple

- Syntax

(original_value = *Title name string at capture*, pattern = *Title name string for matching*, rule = *rule*)

- Predefined constants that can be specified in "rule"

Table 5-2. window_title rules

Predefined constant	Description
$WindowRule.Unspecified	Not specified
$WindowRule.ExactMatch	Match
$WindowRule.PartialMatch	Partial match
$WindowRule.AtFirst	Prefix match
$WindowRule.AtLast	Suffix match
$WindowRule.Regex	Regular expression

5.3.2 window_class const tuple

- Syntax

(original_value = *Class name string at capture*, pattern = *Class name string for matching*, rule = *rule*)

- Predefined constants that can be specified in "rule"

Table 5-3. window_class rules

Predefined constant	Description
$WindowRule.Unspecified	Not specified
$WindowRule.ExactMatch	Match

5.3.3 process_name const tuple

- Syntax

(original_value = *Process name string at capture*, pattern = *Process name string for matching*, rule = *rule*)

- Predefined constants that can be specified in "rule"

Table 5-4. process_name rules

Predefined constant	Description
$WindowRule.Unspecified	Not specified
$WindowRule.ExactMatch	Match

5.3.4 window_size const tuple

- Syntax

(original_value = *Window size string at capture*, pattern = *Window size string for matching*, rule = *rule*)

The window size string at capture or for matching is a string in which the width and the height are connected by a comma. Although it has the same notation as a comma separated number, it is a string of two numeric values, and cannot be treated as one numeric value here.

- Predefined constants that can be specified in "rule"

Table 5-5. window_size rules

Predefined constant	Description
$WindowRule.Unspecified	Not specified
$WindowRule.Equal	Match
$WindowRule.GTE	Greater than or equal to
$WindowRule.LTE	Less than or equal to

5.3.5 Example of the window match rule part

```
Window_rule "Untitled-Notepad" [comment = ""] = (
    window_title = (original_value = "Untitled - Notepad", pattern = "Untitled - Notepad", rule = $WindowRule.ExactMatch),
    window_class = (original_value = "Notepad", pattern = "Notepad", rule = $WindowRule.ExactMatch),
    process_name = (original_value = "notepad.exe", pattern = "notepad.exe", rule = $WindowRule.ExactMatch),
    window_size  = (original_value = "818,388", pattern = "818,388", rule = $WindowRule.Unspecified)
)
```

5.4 Main part

- Syntax

```
main Preamble {
        Sequence of statements
}
```

- Description

This is the part that corresponds to the start to end of a scenario in the flowchart.
The statements written in the main part will be executed in order.
There are no valid attributes in the preamble of the main part.

5.5 Floating part

- Syntax

```
floating Preamble
{
    tag Preamble
    ( tag_comment = Comment string, target = Constant expression, area = Relative position );
}

or

floating Preamble
{
    Sequence of statements
}
```

- Description

In the floating part, you can write one sticky note (tag) or multiple statements.

To associate a sticky note with a node, write the ID attribute in the preamble of the node and specify the ID number in the target attribute of "tag."
The ID number must be an integer, an empty string, or a string that becomes an integer.
If an empty string is specified as an ID number, the sticky note is interpreted as an independent one which has no association

Both the tag_comment attribute and target attribute are optional, and it is assumed that an empty string is specified when omitted.

A “Relative position” of the sticky note may be specified in the area attribute. It must be one of the “Relative position”s in the following table.

The specification to the area attribute may be omitted. In this case, it is interpreted as if the absolute position is specified.

On the flowchart of WinActor, no pull-down menu corresponding to the area attribute exists because the relative position is determined automatically using the position of the sticky note.

Table 5-6. Relative positons of a sticky note

Relative position	Description
$TAG.TopLeft	Upper left
$TAG.TopCenter	Upper middle
$TAG.TopRight	Upper right
$TAG.CenterLeft	Middle left
$TAG.CenterCenter	Middle
$TAG.CenterRight	Middle right
$TAG.BottomLeft	Lower left
$TAG.BottomCenter	Lower middle
$TAG.BottomRight	Lower right

When there are multiple statements, the compiler creates a group node to combine the statements.

Write a tab (tab_id_ref) and node position (x, y) in the flowchart in the preamble of "floating."

For "tab_id_ref," give a string specified in "tab_id" of the flowchart information part.

You can write a name (name) and comment (comment) in the preamble of "tag."

In the preamble of “floating” that has statements, you can specify invisibility (TagVisible) of the sticky note in addition to a tab and a note position.

Write “TagVisible = false” in the preamble to make it invisible.

- Example of the floating part

```
floating [x = 216, y = 37.5, tab_id_ref = 0]
{
    tag [name = "Sticky note", comment = "Sticky note of TimerWait"]
    (tag_comment = "Independent wait", target = 80);          // target = "80" is also acceptable
}

floating [ID = 80, x = 1305, y = 6, tab_id_ref = 0]
```

```
{
    WinActor.TimerWait [name = "Independent place wait", comment = ""]
    (    mode = $TimerWait.Sleep,
         timeout = 10,
         date_format = $TimerWait.ScenarioInfoDateFormat,
         timezone = $TimerWait.ScenarioInfoTimeZone
    );
}
```

5.6 Subroutine part

- Syntax

```
sub Subroutine name Preamble
    localvars( Sequence of variable names ) ,
    chkempty( true or false )
{
   Sequence of statements
}
```

- Description

Write the subroutine name with a string or an identifier. Duplicate names will result in an error. In the preamble, write the tab (tab_id_ref attribute) and node position (x attribute, y attribute) in the flowchart, and the folded state (isclosed attribute, value is true or false). The name attribute has no effect in the preamble. Write “TagVisible = false” in the preamble to make the associated sticky notes invisible.

"chkempty(true or false)" is optional. When omitting, do not write the comma immediately before it.
Variable names in "localvars" must be declared as variables in the variable part.
After the execution of the subroutine, the variable values will be restored to the values before the execution.

In some libraries, the sequence of statements for the subroutine is hidden. The hidden sequence of statements will not be displayed in WSS and cannot be changed.

5.7 WinWatcher part

- Syntax

```
Rules = (
    (window_rule_ref = Window_rule WinID name, WinWatcher action), ...
)
```

- WinWatcher actions

One of the following three:

Table 5-7. WinWatcher actions

WinWatcher action	Description
throw(exception name)	Raises an exception. Write an exception name with a string.
subref(subroutine name)	Runs a subroutine. Write a subroutine name with a string or an identifier.
error	Stops the scenario.

- Description

WinID name must be declared in the window match rule part (Window_rule).
The WinWatcher action is executed when a window that meets the window match rule is displayed.
"Rules" can contain multiple WinID name and WinWatcher action pairs.

- Example of the WinWatcher part

```
Rules = (
    (window_rule_ref = "Warning", throw("Raise warning ")),
    (window_rule_ref = "Enter network credentials", subref("Sound a buzzer")),
    (window_rule_ref = "Restricted", error)
)
```

5.8 EventWatcher part

- Syntax

```
Events = (
    EventWatcher name Preamble = (
        ttrigger = Event trigger condition,
        Event trigger condition parameters,
        Call action parameters,
        fromt_the_start = true or false    // Whether to watch from the start or not
    ), ...
```

- Event trigger condition

Specify one of the followings for the event trigger condition.

Table 5-8. Event trigger conditions

Option	Description
$EVENT.UpdateFile	Update of a specific file
$EVENT.UpdateFolder	Update of a specific folder
$EVENT.SpecifiedTime	Time (specific time)
$EVENT.Monthly	Time (monthly)
$EVENT.Weekly	Time (weekly)
$EVENT.Everyday	Time (every day)
$EVENT.Hour	Time (every hour)
$EVENT.Minute	Time (every minute)
$EVENT.WindowState	Status of the window
$EVENT.Mail	Mail reception

- Event trigger condition parameters

Parameters for each event trigger condition is described below.

Update of a specific file

```
    path = File path to watch event,
```

Update of a specific folder

```
    path = Folder path to watch event,
```

Time (specific time)

```
    datetime = time,      // yyyy/MM/dd HH:mm:ss format
```

Time (monthly)

```
type = Identifier of a monthly event type,
day = day,              // dd format, effective only when $EVENT.DAY is selected for the type
hour = hour,            // HH format
minute = minuute,       // mm format
```

Specify one of the followings as an identifier of a monthly event type.

Table 5-9. Identifiers of monthly event types

Option	Description
$EVENT.DAY	Every month (specified day)
$EVENT.STARTDAY	Beginning of every month
$EVENT.LASTDAY	End of every month

Time (weekly)

```
day_of_the_week = Identifier of a day of the week,
hour = hour,            // HH format
minute = minuute,       // mm format
```

Specify one of the followings as an identifier of a day of the week.

Table 5-10. Identifiers of days of the week

Option	Description
$EVENT.Mon	Monday
$EVENT.Tue	Tuesday
$EVENT.Wed	Wednesday
$EVENT.Thu	Thursday
$EVENT.Fri	Friday
$EVENT.Sat	Saturday
$EVENT.Sun	Sunday

Time (every day)

```
hour = hour,            // HH format
minute = minuute,       // mm format
```

Time (every hour)

```
interval = interval hours,    // 1 - 99
minute = minuute,             // mm format
```

Time (every minute)

```
interval = interval minutes,    // 1 - 120
```

Status of the window

```
window_rule_ref = WinID name,
win_state = expected status,
```

See “8.3.4 Wait for Window Status” for these parameters.

Mail reception

No parameter.

- Call action parameters

Parameters for each call action type is described below.

Call subroutine

```
callsub String or identifier Preamble ( Sequence of expressions ),
```

See “6.12 Call subroutine statement” for these parameters.

Call scenario

```
call_scenario Preamble (
    file = variable name or filename,
    call_vars = (calee's variable name1 = initial value1, …),
    return_vars = (variable name1 that receives calee's value = initial value1, …),
    return_value = variable name that receives the return value
),
```

See “6.13 Call scenario statement” for these parameters

- Description

Specify EventWatcher name as a string or an identifier.
Specify ‘Event trigger condition,’ ‘Event trigger condition parameters,’ ‘Call action parameters,‘ and ‘Whether to watch from the start.’

An error occurs when the specified EventWatcher names is duplicated.

An error occurs when the specified Event trigger condition parameters are not in accordance with the specified Event trigger condition.

- Example of the EventWatcher part

```
Events = (
    “EventWatcher name” [comment = “Comment”] = (
        trigger = $EVENT.UpdateFile,
        path = @"C:\temp\WinActor\WathcerData.txt",
        ‘return val’ = callsub “Subroutine group” [name = “Event list:Event watcher”] (),
        from_the_start = false
    )
)
```

5.9 Breakpoint information part

- Syntax

```
Breakpoint_info = (
    ( id = Constant expression, enable = true or false ), ...
)
```

- Description

You can write multiple breakpoint information.
For the breakpoint information, specify an ID and whether the breakpoint is enabled or disabled.
The ID is the ID number specified in the ID attribute of the preamble of a node where the breakpoint is set.

- Example of the breakpoint information part

```
Breakpoint_info = (
    ( id = 88, enable = true )
)
```

5.10 Scenario information part

- Syntax

```
Scenario_info = (
    creator      = Creator string,
    contact      = Contact string,
    expiration   = Expiration string,
    remarks      = Remarks string,
    dataupdate_change  = true or false,
    ignore_datawrite_error = true or false,    // If true, an error when writing a data list will be ignored.
    variable_limit  = true or false,           // If true, the number of characters in variable values will be
                                                  limited.
    save_ignore_exec = true or false,          // If true, the run/skip node status will be saved.
    user_dictionary_enable  = true or false, // If true, the user translation dictionary will be used when
                                                  running a scenario.
    wss_integer_arithmetic_only = true or false,
                                               // If true, all four arithmetic operations appeared in
                                                  expressions and constant expressions are treated as
                                                  integer arithmetic operations.
    wait_setting       = Timeout option,
    wait_timeout       = Timeout period,       // milliseconds
    percent_variable = true or false,
    use_webdriver    = true or false,
    collect_healing_info = true or false,
    alter_property_path = true or false
)
```

- Description

This is the part corresponding to the scenario information window of WinActor.
For the attributes that can be specified in the above syntax, the value of scenario information can be changed in WSS.
“wss_integer_arithmetic_only” can be used only in WSS. It does not exist in the scenario information of WinActor.

For “wait_setting”, give an option to indicate a source of the timeout period. The option is one of the followings.

Table 5-11. Options for the source of the timeout period

Option	Description
$SCENARIO_INFO.WaitSettingOption	Timeout period in “Option” window
$SCENARIO_INFO.WaitSettingScenario	Timeour period in “Scenario information” window

“wait_setting” is optional. ‘$SCENARIO_INFO.WaitSettingOption’ is the default.

For “wait_timeout”, give a timeout period in milliseconds or the variable that store a timeout period. The timeout period should be in the range of 100 to 3,600,000.

“wait_timeout” is optional. The default value is 10,000.

When ‘$SCENARIO_INFO.WaitSettingOption’ is specified for “wait_setting,” the value specified for “wait_timeout” is not used.

For “percent_variable,” specify true or false. If omitted, it is assumed to be false.

For “use_webdriver,” specify true or false. If omitted, it is assumed to be true on the script created with WinActor before Ver.7.4, and to be false on the script created with WinActor Ver.7.4 or later.

For “collect_healing_info,” specify true or false. If omitted, it is assumed to be false,

For “alter_property_path,” specify true or false. If omitted, it is assumed to be false. This setting is effective only when “collect_healing_info = true” is specified.

- Example of the scenario information part

```
Scenario_info = (
    creator = "User A",
    contact = "User A",
    expiration = "2030/09/30 00:00:00",
    remarks = "",
    dataupdate_change = true,
    ignore_datawrite_error = false,
    variable_limit = true,
    save_ignore_exec = false,
    user_dictionary_enable = false,
    percent_variable = false,
    use_webdriver = false,
    collect_healing_info = false
)
```

5.11 Image part

- Syntax

```
Images = (
   Image declaration ,...
)
```

- Description

The image part lists reference images held by WinActor.
Reference images are acquired by WinActor. They will not be recognized by WinActor when they are written only in WSS.

An error will occur if the image IDs of the image declaration are duplicated.

5.11.1 Image declaration

- Syntax

```
Image ID = ( Attribute = Attribute value , ...)
```

- Attributes

Table 5-12. Image declaration attributes

Attribute	Description
name	Image name string
size	Image size (percentage) when the image was captured. Numeric value from 0 to 100.
width	Numeric value of the image width
height	Numeric value of the image height

5.11.2 Example of image declarations

```
Images = (
      img_20190613104343897 = (name = "Untitled-Notepad", size = 50, width = 409, height = 194),
      img_20191210154248256 = (name = "start-GoogleSearch-InternetExplorer", size = 50, width = 562,
height = 531)
)
```

5.12 Flowchart information part

- Syntax

```
Flow_divide_info = (
    Tab name string = ( seq = Constant expression, tab_id = String ) , ...
)
```

- Description

An error will occur if the tab name strings are duplicated.
"tab_id" should not be duplicated.

"tab_id" = "0" is mandatory.

"tab_id" is referenced from "tab_id_ref" of the preamble of the floating part or the subroutine part. An error will occur if the reference destination of "tab_id_ref" cannot be found.

- Example of the flowchart information part

```
Flow_divide_info = (
     "Main" = (seq = 1, tab_id = "0"),
     "NewTab_1" = (seq = 2, tab_id = "1")
)
```

5.13 Word dictionary part

- Syntax

```
Translation = (
    (Country identification string, Country identification string, ... ),
    (Word string,   Word string, ...),
    ...
)
```

- Description

This part sets up a word dictionary for each language.
The word dictionary can be set only in WinActor itself. The word dictionary part written in WSS will not be reflected in WinActor itself.

First, write a const tuple of language type strings as a title.
Subsequently, write multiple const tuples with corresponding words.
The number of elements in all tuples must be the same.

- Example of the word dictionary part

```
Translation = (
    ("ja_JP", "en_US"),
    ("テキスト", "text"),
    ("入出力", "io"),
)
```

6. Statement

6.1 Description

Letters .,; () {} are syntax elements. The italic parts are explained separately.

The statement preamble is optional.

Multiple statements can be written in "Sequence of statements." The number of statements can be 0.

"Expression;" does not become a statement.

There is no empty statement, and only ";" is not allowed as a statement.

6.2 Group statement

- Syntax

```
Group Preamble
{
    Sequence of statements
}
```

This is used to organize statement sequences.

When you write a sequence of statements in the floating part, the compiler puts them together in a group statement.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.3 if statement

- Syntax

```
if ( Conditional expression ) Preamble
then Preamble
{
    Sequence of statements
}
else Preamble
{
    Sequence of statements
}
```

"else preamble {}" of the "else" part is optional.

Only the name attribute is valid for the preamble of the "then" part and the "else" part.

To make associated sticky notes invisible, write “TagVisible = false” in the first preamble.

6.4 while statement

- Syntax

```
while Preamble
Loop condition
( Counter Identifier )
{
   Sequence of statements
}
```

"(Counter identifier)" that specifies a counter variable is optional.

No special identifier can be specified for the counter. Variables declared as constants cannot be specified either.

The anonymous identifier '' can be specified for the counter.

"while" and "Counter" are case insensitive.

If the attribute "isclosed_body" is set to true in the preamble, the while node will be displayed with the body part closed.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.4.1 Loop condition

- Syntax

One of the following eight formats:
For details, see the "Pre-Test Loop" node in "WinActor Operation Manual."

Table 6-1. Loop condition formats

Format	Description
(conditional expression)	Repeats the loop until the conditional expression becomes false. Details of the conditional expression are described later.
(true)	Always represents a true conditional expression. Keeps to repeat the loop.
(false)	Always represents a false conditional expression. Never being in a loop.

Format	Description
(Start = expression, End = expression)	Repeats the loop within the specified range of numbers.
(File = string or variable name)	Specifies an Excel or a CSV file and repeats the loop for the number of data.
(DBSource = string or variable name , User = string or variable name , Password = string or variable name , Table = string or variable name)	Repeats the loop for the number of data records in the database.
(Template_And_Data = variable name, Iterate = true or false, IterateOver = expression, IsUpdate = true or false, UpdateTo = variable name)	Repeats the loop for the number of data using template and data. 'IterateOver' is effective only when "Iterate = true," and 'UpdateTo' is effective only when "IsUpdate = true."
(Json_Object = variable name or JSON array string Key = expression, KeyOut = variable name, ValueOut = variable name)	Repeats the loop obtaining keys and values from JSON object or JSON array.

Following are common loop condition syntax for the while statement and dowhile statement.

Enclose the loop condition in parentheses.

true, false, Start, End, File, DBSource, User, Password, Table, Template_And_Data, Iterate, IterateOver, IsUpdate, UpdateTo, Json_Object, Key, KeyOut, and ValueOut are case insensitive.

6.5 dowhile statement

- Syntax

```
dowhile Preamble
Loop condition
( Counter Identifier )
{
   Sequence of statements
}
```

"(Counter identifier)" that specifies a counter variable is optional.

No special identifier can be specified for the counter. Variables declared as constants cannot be specified either.

The anonymous identifier '' can be specified for the counter.

"dowhile" and "Counter" are case insensitive.

If the attribute "isclosed_body" is set to true in the preamble, the dowhile node will be displayed with the body part closed.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.5.1 Loop condition

Same as "6.4.1 Loop condition."

6.6 switch statement

- Syntax

```
switch Preamble
Sequence of case statements
Default statement
```

The default statement is optional.

When an expression including calculation is given to a case statement, the compiler replaces "switch" with an "if-then-else" statement, and the representation on the flowchart changes.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.6.1 Case statement

- Syntax

```
Case ( Conditional expression ) Preamble
{
    Sequence of statements
}
```

Conditional expression is required.

"Case" is case insensitive.

The preamble is optional.

Only the name attribute is valid for the preamble.

6.6.2 Default statement

- Syntax

```
Default
{
    Sequence of statements
}
```

No preamble can be added to the default statement.

"Default" is case insensitive.

6.7 try statement

- Syntax

```
try Preamble
{
    Sequence of statements
}
Sequence of catch statements
```

At least one catch statement is required.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.7.1 catch statement

- Syntax

```
catch Exception handling name string Preamble
{
    Sequence of statements
}
```

The preamble is optional. There are no valid attributes.

6.8 return statement

- Syntax

```
return ( Expression ) Preamble ;
```

The return statement can be used only within a subroutine block defined in the subroutine part.

"Expression" is optional, but "(" and ")" cannot be omitted.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.9 scenario return statement

- Syntax

```
scenario_return ( Expression )   Preamble ;
```

"Expression" and “Preamble” are optional.

The value of the “Expression” is the return value to the statement “call_scenario” in the caller’s scenario.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

If this statement is not placed in the main part, a warning occurs.

6.10 break statement

- Syntax

```
break Preamble ;
```

The break statement can be used only within a "while" or "dowhile" block.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.11 continue statement

- Syntax

```
continue Preamble ;
```

The continue statement can be used only within a "while" or "dowhile" block.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.12 Call subroutine statement

- Syntax

```
callsub String or identifier Preamble ( Sequence of expressions ) ;
```

Define a subroutine corresponding to "String or identifier".

"callsub" is optional.

"Sequence of expression" is optional, but "(" and ")" cannot be omitted.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.13 Call scenario statement

- Syntax

```
Call_scenario Preamble (
   file = variable name of filename
   call_vars = (callee’s variable name1 = initial value1, …),
   return_vars = (variable name1 that receives callee’s value = initial value1, …),
   return_value = variable name that receives the return value
) ;
```

The value of “call_vars” is pairs of a callee’s variable name and its initial value if the pairs exist. The “call_vars” is optional.

If a callee’s variable name of a pair does not exist, the pair is ignored.

Not to specify an initial value, write an anonymous identifier ‘’ (two single quotes) as a value.

To specify a value without a variable name, write “ ‘’ = value” however a warning occurs when such a script is loaded.

The value of “return_vars” is names of variables to be returned when the callee’s scenario ends. The “return_vars” is optional.

The return value of the callee will be stored in the variable specified for the “return_value.”

The value is returned in the scenario_return statement in the callee’s scenario.

The return value is not stored in the specified variable when this statement appears in an assignment statement or in an expression. When the assign statement or expression

appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

The “return_value” is optional.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

Example

```
call_scenario [name = "Call Scenario File", comment = ""]
(
    file = @"C:\Users\user\Documents\sub.ums7",
    call_vars = (name = "user"),
                    //Set "user" to the “name” in the scenario “sub.ums7” and launch it
    return_vars = (ret)          //Receive the value of the ”ret” in the scenario “sub.ums7
);


result = call_scenario [name = "Call Scenario File", comment = ""]
                                                    // Assign the result set with scenario_return
(
    file = @"C:\Users\user\Documents\sub2.ums7",
    call_vars = (),        // Omit variables and values
    return_vars = ()       // Omit variables
);
```

6.14 Adapter action statement

- Syntax

```
Adapter action ;
```

Calls an adapter action.

In the case of an action that returns a value, the value will be discarded unless the return destination of the value is set in the action parameter.

When calling an action that returns a value, it is recommended to place an adapter action on the right side of an assignment statement. When the assign statement appears in the WSS output of a flowchart, the property settings for storing the return value is omitted.

See the chapter on adapter actions.

6.15 Assignment statement

- Syntax

Identifier = *Expression Preamble* ;

Assigns a value of an expression to a variable.

The identifier on the left side of "=" is one of the following.

Table 6-2. Identifiers in the assignment statement

Identifier
Identifier declared as a variable in the variable part
Read-write special identifier

If you specify an ID (such as for a breakpoint) to both the preamble of an assignment statement and the preamble of the element in the expression (such as an adapter action), the both IDs will be effective, and there will be no optimization of reducing assignment operations.

Specifying an ID only for the preamble of the element in the expression may reduce the number of nodes to be generated.

To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional.

6.16 Four arithmetic operations

The four arithmetic operations can be written freely.

The representation on the flowchart may change because the compiler adds the necessary nodes when converting WSS into nodes.

7. Expression

An expression is a combination of one or more factors with the four arithmetic operators of +, -, *, /, +^, -^, *^, and /^..

You can place unary operators of +, -, +^, and -^ in front of the factor.

The priority of operators is as follows.

Table 7-1. Priority of operators

No.	Operator
1	Unary operator + - +^ -^
2	* / *^ /^
3	+ - +^ -^

When a factor at the time of execution is a simple string literal, the operation by the unary operator and the four arithmetic operators cannot be performed.

However, if the factor is a string literal that can be interpreted as a number, these operations can be performed.

There is no operation to obtain a Boolean value.

Operators of +^, -^, *^,and /^ are used in integer arithmetic. They correspond to the nodes of four arithmetic operations with the “Calculate as an integer and truncate the result numbers beyond the decimal point.” checked.

They can be used with +, -, *, and / by mixture.

If any operand of the operator is not an integer, a runtime error occurs. A comma separated number is not an integer thus a runtime error occurs if it is an operand of the operator..

When unary operators +^, -^ are placed before a factor, they are treated as “(0 +^ *factor*)”, “(0 -^ *factor*)” respectively and checked for whether they are integers or not.

When even number of unary operators of integer arithmetic are placed like “-^ -^ *factor*”, they are treated as “(0 +^ *factor*)”.

Even if a factor is a comma separated number with full-width commas or noted with full-width digits, the result of an operation is a number noted with half-width digits.

7.1 Factor

- Syntax

A factor is one of the following:

Table 7-2. Factor syntax

Factor	Remarks
Integer	
Float	
String	Unary operation and four arithmetic operations cannot be applied.
Identifier	
TRUE	Interpreted as the string "TRUE." Unary operation and four arithmetic operations cannot be applied.
FALSE	Interpreted as the string "FALSE." Unary operation and four arithmetic operations cannot be applied.
Adapter action with return value	The property settings for the returned arguments' list are ignored. In the WSS output, the property settings for storing the return value are omitted.
Call subroutine with return value	
Call scenario with return value	
(expression)	

The identifier is a variable, constant, predefined constant, special variable, or anonymous identifier.

Anonymous identifiers and identifiers whose values are string literals cannot perform the unary operation and four arithmetic operations.

7.2 Constant expression

A constant expression is an expression that is calculated to a constant when compiling. The result may be a Boolean value.

It consists of one or more constant factors combined with a binary operator for constant expressions.

String literals that can be interpreted as numbers can be included in constant expressions. However, if the operands on both sides of a comparison operator are string literals, they are not interpreted as numbers.

The operands of && (and) and || (or) must be Boolean values on both sides. Both sides will be evaluated.

The operands of ~ (regular expression match) and !~ (regular expression unmatch) must be string literals on both sides.

The operands of * / + - must be numeric values or string literals that can be interpreted as numbers on both sides.

The operands of *^ /^ +^ -^ must be integer values on both sides. If either of them has a value beyond the decimal point, an error occurs. A comma separated number is not an integer value thus an error occurs if it is included in the operands.

The operands of == != >= > < <= must be numeric values on both sides or string literals on both sides.

An operations on a number and a string literal is not possible.

When both sides are string literals, none of them is interpreted as a number even when both of them can be interpreted as numbers. For example, the operands in “2000” < “300” are regarded as strings on both sides, and the expression is evaluated to be true. The operands in “2000” < 300 are interpreted as numbers on both sides, and the expression is evaluated to be false.

If both sides are string literals, == and != will be case sensitive and compared with exact match.

>= > < <= are case insensitive and compared lexicographically.

7.2.1 Binary operators for constant expressions

The order of priority (highest to lowest) is as follows.

Table 7-3. Binary operators for constant expressions

No.	Binary operator	Remarks
1	* / *^ /^ &&	
2	+ - +^ -^ ||	You can place unary operators of + - ;^ -^ in front of the factor.
3	== != >= > < <= ~ !~	

7.2.2 Constant factors

- Syntax

A constant factor is one of the following:

Table 7-4. Constant factor syntax

Constant factor	Remarks
Integer	
Float	

Constant factor	Remarks
String	Unary operation cannot be applied.
Identifier	
TRUE	Interpreted as the string "TRUE." + - unary operation cannot be applied.
FALSE	Interpreted as the string "FALSE." + - unary operation cannot be applied.
strcmp(constant expression, constant expression)	The constant expression must be a string literal. Case-sensitive comparison, and the result is "true" or "false".
strcasecmp(constant expression, constant expression)	The constant expression must be a string literal. Case-insensitive comparison, and the result is "true" or "false".
! constant factor	The constant factor must be a Boolean value.
(constant expression)	

The identifier is a constant, predefined constant, read-only special variable, or anonymous identifier.

Anonymous identifiers cannot be operated.

For undefined identifiers, a warning will be output and the value will be treated as the integer zero, and the process will continue.

7.3 Conditional expression

A conditional expression consists of one or more conditional expression factors combined with && or ||.

Priority is given to &&, but it is recommended to enclose it in parentheses as much as possible.

The result of a conditional expression is "true" or "false."

7.3.1 Binary operators for conditional expressions

A binary operator for conditional expressions is one of the following:

Table 7-5. Binary operators for conditional expressions

Binary operator
== != >= > < <= ~ !~

7.3.2 Conditional expression factors

The result of a conditional expression factor is "true" or "false."

- Syntax

A conditional expression factor is one of the following:

Table 7-6. Conditional expression factor syntax

Conditional expression factor	Remarks
istrue(expression)	The result is "true" or "false."
isfalse(expression)	The result is "true" or "false."
strcmp(expression, expression)	The expression must be a string literal. Case-sensitive comparison.
strcasecmp(expression, expression)	The expression must be a string literal. Case-insensitive comparison.
Expression Binary operator for conditional expression Expression	
TRUE	
FALSE	
! conditional expression factor	
(conditional expression)	

8. Adapter actions

Adapter actions correspond to nodes in the action category.

An adapter action that returns a value can be included in an expression.

See "4.6 Structure" for general notation for adapter parameter lists.

- Syntax

WinActor.*Action name Preamble Adapter parameter list*

"Action name" is case insensitive.

For an action that returns a value, specify a name of a variable to which the value is to be returned in a specific attribute name (such as value).
If an action that returns a value is used in an expression (including an assignment statement), the return destination specified by the attribute name will be ignored. In this case, the attribute name of the return destination can be omitted. Thus, when the assign statement or expression appears in the WSS output of a flowchart, the attribute name for the return value is omitted.
To make associated sticky notes invisible, write “TagVisible = false” in the preamble.

The preamble is optional except for some parts.

The following shows the adapter parameter list for each action.

8.1 Automatic recording

8.1.1 Event recording – Click

This is an operation of clicking a button, check box, or radio button.

```
WinActor.ClickWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

"window_rule_ref," "control," "capture" are required.

For "Window_rule WinID name," write "WinID name" of the window match rule part. (The same applies to the actions described below.)

"<check>" attached to the attribute of "control" is for specifying check box status in the Details tab displayed in the property of a node recorded in the Event mode.

<true> for checked and <false> for unchecked.

For "instance" of "control," specify a serial number assigned to the control with a number or a variable that stores a number. (The same applies to the actions described below.)

For "text" of "control," specify a string displayed in the control with a string or a variable that stores a string. (The same applies to the actions described below.)

The "position" of "control" is a variable or a string or (x = constant expression, y = constant expression). (The same applies to the actions described below.)

When you do not specify the position, write it as a string "" or the constant expression as "", such as (x = "", y = "").

When you specify it with a variable or a string, write x and y coordinates combined with a comma. (Example: "100,200")

Table 8-1. Sources of the timeout period

Item type	Description
$WIN32.WaitSettingOption	The timeout is set in the “Option” dialog.
$WIN32.WaitSettingScenario	The timeout is set in the “Scenario information” window.
$WIN32.WaitSettingNode	The timeout is set for “wait_timeout” in the “Property” pane.

“wait_setting” is optional. ‘$WIN32.WaitSettingScenario’ is the default.

For “wait_timeout”, give a timeout period in milliseconds or the variable that store a timeout period. The timeout period should be in the range of 100 to 3,600,000. If the specified period is out of the range, a warning message is displayed, and the nearer of the two values 100 and 3,600,000 is used instead.

If an anonymous identifier ‘’ (two single quotes) is specified as a variable name, 10,000 is used as the timeout period.

“wait_timeout” is optional. The default value is 10,000.

When ‘$WIN32.WaitSettingOption’ or ‘$WIN32.WaitSettingScenario’ is specified for the “wait_setting,” the value specified for “wait_timeout” is not used. The “wait_timeout” property of a node is output to the .wss file from WinActor only when ‘Use this ”Property”’ is selected for “Timeout setting” in the “Property” pane of the node.

For "imageid" of "capture," specify an image ID in the image declaration of the image part.

"mouse coordinate X" and "mouse coordinate Y" of "capture" are numerical constants. When you do not specify the coordinates, write them as "".

8.1.2 Event recording – Set Text

This is an operation to set a string in a text box.

```
WinActor.SetTextWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Expression,     // Text string to set
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

If the expression of "value" results in a number, it is regarded as a string.

When you do not set an expression, specify the anonymous identifier '' (two single quotes) in "value."

If you specify "" (two double quotes), it means that you specify an empty string.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.3 Event recording – Select Item in List

This is an operation to select an item in a list box or combo box.

```
WinActor.SelectListWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Expression or variable name,
    kind             = Item type,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "kind," specify the item type $SelectListWin32.index or $SelectListWin32.text.

Table 8-2. Item types in the Select Item in List action

Item type	Description
$SelectListWin32.index	Selects an item in a list by specifying a zero-based index
$SelectListWin32.text	Selects an item in a list by a string displayed in the list

For "value," specify an expression that results in a value for selecting a list box or a variable name that stores a value.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.4 Event recording – Select Tab

This is an operation to switch tabs.

```
WinActor.SelectTabWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Expression or variable name,
    kind             = Item type,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "kind," specify the item type $SelectTabWin32.index or $SelectTabWin32.text.

Table 8-3. Item types in the Select Tab action

Item type	Description
$SelectTabWin32.index	Selects a tab by specifying a zero-based index
$SelectTabWin32.text	Selects a tab by a string displayed on the tab

For "value," specify an expression that results in a value for selecting a tab or a variable name that stores a value.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.5 Event recording – Emulate

This is automatic operations to emulate a sequence of mouse click positions and keyboard operations.

```
WinActor.EmulationWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    action           = Sequence of actions,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "action," write multiple mouse operations, keyboard operations, and waits. The actions will be executed in the order.

The sequence of actions is written as follows:

Table 8-4. Sequence of actions for emulation

Sequence of actions	Description
Action	When writing only one action
(Action, ...)	When writing multiple actions, separate them with commas and enclose them in parentheses.

The following three types of actions are available. These can be mixed.

Table 8-5. Emulation actions

Action
Mouse action
Key action
Wait

Mouse action

```
@(Mouse, button, movement, X-coordinate, Y-coordinate, origin, X_D/P, Y_D/P, Scale)
```

Table 8-6. Mouse actions for emulation

Mouse action	Description	
Button	One of the following:	
	L	Left button
	R	Right button
	M	Middle button
Movement	One of the following:	
	DOWN	Button down
	UP	Button up
	MOVE	Move
	DBL	Button double-click
Origin	One of the following: (case insensitive)	
	LEFTTOP	Upper left origin
	RIGHTTOP	Upper right origin
	LEFTBOTTOM	Lower left origin
	RIGHTBOTTOM	Lower right origin
X_D/P Y_D/P	Select a method to specify X and Y coordinates. One of the following: (case insensitive)	
	D	Direct (in pixels)
	P	% (percentage to the width or height of the window)
Scale	Scale factor of captured image	1.0 for equal magnification optional

Key action

```
@(Key, key code, UP/DOWN)
```

Wait

```
@(Wait, wait time)
```

The unit of wait time is milliseconds

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

Example

```
Var_group (
   const DefaultWaitTime = 2*1000 [comment = "2sec"]
)

// Mouse operation and wait
```

```
action = ( @(Mouse, L, DOWN, 796, 56, LEFTTOP, D, D),
           @(Mouse, L, UP, 796, 56, LEFTTOP, D, D),
           @(Mouse, NON, MOVE, 799, 100, LEFTTOP, D, D),
           @(Wait, 1000)),

// Keyboard operation and wait
action = ( @(Key, 18, DOWN),
           @(Key, 115, DOWN),
           @(Key, 115, UP),
           @(Key, 18, UP),
           @(Wait, (DefaultWaitTime + 2 * 500))),      // Enclose constant expressions in parentheses

// Write only one action
action = @(Wait, 300),
```

8.1.6 Event recording – Get String

```
WinActor.GetTextWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Variable name to receive the result,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

The result is returned to the variable specified in "value." Specify a writable variable.

If WinActor.GetTextWin32 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

Example

```
Var_group (
  var01 = 0 [comment = "work variable"],
  ret01 = 0 [comment = "for return"]
)
// Assign the result to a variable.
ret01 = WinActor.GetTextWin32 [name = "Get String (WIN32)", comment = "get text"]
(
    window_rule_ref  = "Untitled-Notepad",
    control          = (instance<true> = var01, text<true> = var01, position<true> = var01),
```

```
        capture             = (imageid = "img_20191115153616376", x = 1197, y = 159)
    );
    // Specify a variable name to return the result (value attribute)
    WinActor.GetTextWin32 [name = "Get String (WIN32)", comment = "get text"]
    (
        window_rule_ref  = "Untitled-Notepad",
        control             = (instance<true> = var01, text<true> = var01, position<true> = var01),
        value               = ret01,
        capture             = (imageid = "img_20191115153616376", x = 1197, y = 159)
    );
```

8.1.7 Event recording – Get Item in List

```
WinActor.GetListWin32 Preamble
    (
      window_rule_ref  = Window_rule WinID name,
      control             = (instance<check> = expression, text<check> = expression, position<check> =
position),
      value               = Variable name to receive the result,
      kind                = Item type,
      wait_setting        = Timeout option,
      wait_timeout        = Timeout period,     // milliseconds
      capture             = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
    )
```

The result is returned to the variable specified in "value."

If WinActor.GetListWin32 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For "kind," specify the item type $GetListWin32.index or $GetListWin32.text.

Table 8-7. Item types in Get Item in List

Item type	Description
$GetListWin32.index	Gets an index of the selected element in the list
$GetListWin32.text	Gets a name of the selected element in the list

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.8 Event recording – Get Check State

```
WinActor.GetCheckWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Variable name to receive the result,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

True or false is returned to the variable specified in "value."

If WinActor.GetCheckWin32 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.9 Event recording – Get Enable/Disable State

```
WinActor.GetEnableWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    value            = Variable name to receive the result,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

True or false is returned to the variable specified in "value."

If WinActor.GetEnableWin32 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.10 Event recording – Get All Items in List

```
WinActor.GetAllListWin32 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (instance<check> = expression, text<check> = expression, position<check> =
position),
    file             = Variable name or filename,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,     // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "file," specify a filename string to output the result or a variable name that stores a filename.

For “wait_setting” and “wait_timeout,” see description in “8.1.1 Event recording – Click.”

8.1.11 UIAutomation

```
WinActor.UIAutomation Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    path             = Control path JSON-format string,
    expand_variable  = true / false,    // whether to expand embedded variables or not
    pattern          = Control pattern,
    action           = Action,
    use_wildcard     = true / false,    // whether to enable ambiguous specification with ‘*’
    wait_setting     = Timeout setting,
    wait_timeout     = Timeout,       // milliseconds
    wait_timeout_period = Wait condition,
    cache_update     = true / false,    // Forced update flag of the cache, Default: false
    wait_retry_max   = Maximum number of retries,
    path_version = $UIA.Version2,        // or $UIA.Version1
    box_center_position_x = Variable to store X coordinate of the element center,
            // Extended Mouse Operation: “Get the coordinates of the element center” only
    box_center_position_y = Variable to store Y coordinate of the element center,
            // Extended Mouse Operation: “Get the coordinates of the element center” only
    box_left_position_x = Variable to store X coordinate of the top-left of the element,
            // Extended Mouse Operation: “Get top-left/bottom-right coordinates of the element” only
    box_top_position_y = Variable to store Y coordinate of the top-left of the element,
            // Extended Mouse Operation: “Get top-left/bottom-right coordinates of the element” only
    box_right_position_x = Variable to store X coordinate of the bottom-right of the element,
            // Extended Mouse Operation: “Get top-left/bottom-right coordinates of the element” only
```

```
    box_bottom_position_y = Variable to store Y coordinate of the bottom-right of the element,
            // Extended Mouse Operation: "Get top-left/bottom-right coordinates of the element" only
    activate_target    = true / false,
                                // whether to activate the target window during execution, Default: true
    control            = Control spec,
    result             = Variable name to receive the result,
    capture            = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
)
```

For "path," specify a target control by a control path expressed in JSON-format string.

As the JSON-format string usually includes many double quotes, it is recommended to write it as a verbatim string literal.

The "path" is required.

To expand variables in the "path" string, include *%variable_name%* in the string and set *true* to the "expand_variable." If the "expand_variable" is not specified, it is regarded as *false*.

For "pattern," specify a control pattern which specifies an operation for the target control.

The "pattern" can be omitted. The control pattern corresponding to the action specified for the "action" is used if "pattern" is omitted.

For the valid pairs of a control pattern and an action, see the "Table 8-11 Adapter parameter list of UIAutomation."

The common prefix "$UIA" is omitted for the identifiers in the table.

For "action," specify an action, which is defined for each control patterns,

For the valid pairs of a control pattern and an action, see the "Table 8-11 Adapter parameter list of UIAutomationtable."

The common prefix "$UIA" is omitted for the identifiers in the table.

For "use_wildcard," set whether to enable or disable the ambiguous specification with '*' in the target control specification of UIAutomation.

Set true to enable the ambiguous specification.

Set false to disable the ambiguous specification.

"use_wildcard" is optional. If omitted, it is assumed to be false, which disables the ambiguous specification.

For “wait_setting,” specify the place to set the timeout, which is the time to wait for the target control and its window to be present after the UIAutomation node is being executed. Select one of the followings as the place.

Table 8-8. Sources of the timeout period

Item type	Description
$UIA.WaitSettingOption	The timeout is set in the “Option” window.
$UIA.WaitSettingScenario	The timeout is set in the “Scenario information” window.
$UIA.WaitSettingNode	The timeout is set for “wait_timeout” in the “Property” pane.

The “wait_setting” is optional. $UIA.WaitSettingNode is the default value.

For “wait_timeout,” specify a timeout period in milliseconds or the variable that store a timeout period. The timeout period should be in the range of 100 to 3,600,000. If the specified value is out of the range, a warning message is displayed, and the nearer of the two values 100 and 3,600,000 is used instead.

If an anonymous identifier ‘’ (two single quotes) is specified as a variable name, 30,000 is used as the timeout period.

The “wait_timeout” is optional. The default value is 30,000.

When ‘$UIA.WaitSettingOption’ or ‘$UIA.WaitSettingScenario’ is specified for “wait_setting,” the setting in “wait_timeout” is not used.

For “wait_timeout_period,” select one of the following waiting conditions.

Table 8-9. Waiting conditions

Waiting condition	Description
$UIA.WaitForWindow	Waiting for the window to be found
$UIA.WaitForControl	Waiting for the target control to be found

The “wait_timeout_period” is optional. The default value is $UIA.WaitForControl.

For “cache_update,” specify whether to update the cache used in the execution of UIAutomation node forcibly.

When true is specified, UIAutomation node updates the cache forcibly and run slowly.

When false is specified, UIAutomatin node avoids the update of the cache as long as possible.

“cache_update” is optional. The default value is false, which means the fast mode.

For “wait_retry_max,” specify the maximum number of retries, which are performed when “unauthorized operation” error occurs, in decimal number.

If 0 is specified, no retry is done.

When a negative value is specified, no limit is set on the number of retries.

The “wait_retry_max” is optional. The default value is 5.

For “path_version,” select one of the following control path format versions. If omitted, it is assumed to be $UIA.Version1.

Table 8-10. Control path format versions

Format version	Description
$UIA.Version1	Version 1, which is compatible with the format of WinActor7.4.4 or earlier
$UIA.Version2	Version 2

The “box_center_position_x” and “box_center_position_y” can be specified only when both “pattern = $UIA.MouseExtensionPattern” and “action = $UIA.GetBoxCenterPosition” are set, otherwise an error occurs. Specify the name of a variable for each of them to store the X and the Y coordinates of the center of the element. These specifications are optional, and an anonymous identifier is used as default.

The ”box_left_position_x,” “box_top_position_y,” “box_right_position_x,” and “box_bottom_position_y” can be specified only when both “pattern = $UIA.MouseExtensionPattern” and “action = $UIA. UIA.GetBoxPositions” are set, otherwise an error occurs. Specify the name of a variable for each of them to store the X and the Y coordinates of the top-left and the bottom-right of the element. These specifications are optional, and an anonymous identifier is used as default.

For the “activate_target,” specify whether to activate the target window during execution.

When true is specified, the target control is operated after the target window is activated.

When false is specified, the target control is operated without activating the target window.

The “activate_target” is optional. The default value is true.

For the “control,” specify parameters necessary to the action as an adapter parameter list.

The “control” can be omitted.

A parameter in the adapter parameter list should be one of three parameters “scroll”, “selection”, and “value”.

Specify each parameter as shown below. For the valid parameters for each action, see “Table 8-11 Adapter parameter list of UIAutomation”.

Invalid parameters will be ignored with showing warning messages at the time of loading.

```
control = (scroll      = (hscroll_amount = Scroll direction and amount identifier,
                          vscroll_amount = Scroll direction and amount identifier),
           selection = (item_index = Expression, item_value = Expression),
           value      = (item_value = Expression, mode = Mode identifier)
          )
```

The result of this actions is returned to the variable specified for the “result”.

The return value will not be stored in the specified variable when this action occurs in an assignment statement or in an expression.

The variable name for the “result” can be omitted. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

Example

```
WinActor.UIAutomation [name = "Select/GetByText/Value", comment = ""]
          (
               window_rule_ref = "Window",
               path = @"""
[{"id":"15","index":"0"}]
""",
               pattern = $UIA.SelectionPattern,
               action = $UIA.SelectItemByText,
               control = (selection = (item_value = 3))
          );

ret = WinActor.UIAutomation [name = "Common/GetLabel", comment = ""]
     (
          window_rule_ref = "title_*Untitled-Notepad",
          path = @"""
[{"id":"Item 5","index":"0"}]
""",
          pattern = $UIA.CommonPattern,
          action = $UIA.GetName,
          control = (),
          capture = (imageid = "img_20200929092507503", x = 445, y = 10)
     );
```

Table 8-11 Adapter parameter list of UIAutomation

pattern Control pattern $UIA.___	action Action $UIA.___	control parameters						result
		scroll		selection		value		
		hscroll_amount	vscroll_amount	item_index	item_value	item_value	mode	
CommonPattern	GetName							○
ExpandCollapsePattern	Expand							
	Collapse							
InvokePattern	Invoke							
ScrollPattern	IsHorizontallyScrollable							○
	GetHorizontalViewportRatio							○
	GetHorizontalViewportSize							○
	HorizontalScroll	○1						
	IsVerticallyScrollable							○
	GetVerticalViewportRatio							○
	GetVerticalViewportSize							○
	VerticalScroll		○1					
	TwoWayScroll	○1	○1					
SelectionPattern	IsMultiSelectable							○
	IsSelectionNeeded							○
	GetSelectionByTexts							○
	GetSelectionByIndexes							○
	GetSelectableItemNum							○
	GetSelectableItems							○
	SelectItemByText				○2			
	SelectItemByIndex			○2				
	IsSelected							○

pattern Control pattern $UIA.___	action Action $UIA.___	control parameters						result
		scroll		selection		value		
		hscroll_amount	vscroll_amount	item_index	item_value	item_value	mode	
SelectionItemPattern	SelectAdditionally							
	Unselect							
	SelectOne							
TogglePattern	Toggle							
	GetToggleState							○
ValuePattern	IsReadOnly							○
	GetValue							○
	SetValue					○3	○4	
MouseExtensionPattern	GetBoxCenterPosition ○5							
	GetBoxPositions ○6							
	MoveToTheElementCenter							
	LeftClickTheElementerCenter							
	RightClickTheElementCenter							
UnknownPattern	Unknown							

1. default $UIA.NoAmount . 2. default 0. 3. default 0.

4. default $UIA.ModeNormal .

5. box_center_position_x, box_center_position_y are effective.

6. box_left_position_x, box_top_position_y, box_right_position_x, box_bottom_position_y are effective.

Identifiers of “Control pattern” and “Action” will be prefixed with “$UIA.”

8.1.12 UIAutomation library

The adapter actions listed on the “Table 8-12” are frequently used operations in UIAutomation. The control patterns and the actions of those adapter actions are fixed.

The function of each adapter action is the same as that of the UIAutomnation adapter action with the same pair of the control pattern and the action.

```
WinActor.Adapter action name Preamble
    (
        window_rule_ref = Window_rule WinID name,
        path = Control path JSON-format string,
        expand_variable = true / false,          // whether to expand embedded variables or not
        use_wildcard = true / false,             // whether to enable ambiguous specification with '*'
        wait_setting  = Timeout setting,
        wait_timeout = Timeout,                  // milliseconds
        wait_timeout_period = Wait condition,
        cache_update = true / false,             // Forced update flag of the cache, Default: false
        wait_retry_max = Maximum number of retries,
        path_version = $UIA.Version2,        // or $UIA.Version1
        control = Control spec,
        result = Variable name to receive the result,
        capture  = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
    )
```

The control pattern and the action set for each adapter action is shown on the table below. Each value of control patterns and actions is fixed to the value on the table, and cannot be specified.

Table 8-12 Control pattern and Action of UIAutomation libraries

Adapter action name	Control pattern	Action
UiaExpandMenu	$UIA.ExpandCollapsePattern	$UIA.Expand
UiaCollapseMenu	$UIA.ExpandCollapsePattern	$UIA.Collapse
UiaClick	$UIA.InvokePattern	$UIA.Invoke
UiaGetItemTextInList	$UIA.SelectionPattern	$UIA.GetSelectionByTexts
UiaGetItemIndexInList	$UIA.SelectionPattern	$UIA.GetSelectionByIndexes
UiaGetAllItemTextInList	$UIA.SelectionPattern	$UIA.GetSelectableItems
UiaSelectItemTextInList	$UIA.SelectionPattern	$UIA.SelectItemByText
UiaSelectItemIndexInList	$UIA.SelectionPattern	$UIA.SelectItemByIndex
UiaSelectTab	$UIA.SelectionItemPattern	$UIA.SelectOne
UiaSelectRadioButton	$UIA.SelectionItemPattern	$UIA.SelectOne
UiaGetText	$UIA.ValuePattern	$UIA.GetValue
UiaSetText	$UIA.ValuePattern	$UIA.SetValue
UiaSetChecked	$UIA.TogglePattern	$UIA.SetChecked

For other parameters, see “8.1.11 UIAutomation.”

8.1.13 UIAutomation dump

```
WinActor.UiaDump Preamble
    (
        window_rule_ref = Window_rule WinID name,,
        output_filename = Filename or Variable name,
        wait_setting      = Timeout option,
        wait_timeout      = Timeout period,     // milliseconds
        capture = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
    )
```

No value is returned.

For "output_filename," specify a filename string to output the dump result or a variable name that stores a filename.

Table 8-13. Sources of the timeout period

Item type	Description
$UIADUMP.WaitSettingOption	The timeout is set in the “Option” dialog.
$UIADUMP.WaitSettingScenario	The timeout is set in the “Scenario information” window.
$UIADUMP.WaitSettingNode	The timeout is set for “wait_timeout” in the “Property” pane.

“wait_setting” is optional. ‘$UIADUMP.WaitSettingScenario’ is the default.

For “wait_timeout”, give a timeout period in milliseconds or the variable that store a timeout period. The timeout period should be in the range of 100 to 3,600,000. If the specified period is out of the range, a warning message is displayed, and the nearer of the two values 100 and 3,600,000 is used instead.

If an anonymous identifier ‘’ (two single quotes) is specified as a variable name, 10,000 is used as the timeout period.

“wait_timeout” is optional. The default value is 10,000.

When ‘$UIADUMP.WaitSettingOption’ or ‘$UIADUMP.WaitSettingScenario’ is specified for the “wait_setting,” the value specified for “wait_timeout” is not used. The “wait_timeout” property of a node is output to the .wss file from WinActor only when ‘Use this ”Property”’ is selected for “Timeout setting” in the “Property” pane of the node.

8.2 Automatic recording (IE)

8.2.1 IE mode recording – Click

```
WinActor.ClickIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,        // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For “wait_setting”, give an option to indicate a source of the timeout period for the IE action while running the scenario. The option is one of the followings.

Table 8-14. Sources of the timeout period

Item type	Description
$IE.WaitSettingOption	The timeout is set in the “Option” dialog.
$IE.WaitSettingScenario	The timeout is set in the “Scenario information” window.
$IE.WaitSettingNode	The timeout is set for “wait_timeout” in the “Property” pane.

“wait_setting” is optional. ‘$IE.WaitSettingScenario’ is the default.

For “wait_timeout,” give a timeout period in milliseconds or the variable that store a timeout period. The timeout period should be in the range of 100 to 3,600,000. If the specified value is out of the range, a warning message is displayed, and he nearer of the two values 100 and 3,600,000 is used instead.

If an anonymous identifier ‘’ (two single quotes) is specified as a variable name, 10,000 is used as the timeout period.

“wait_timeout” is optional. 10,000 is the default value.

When either ‘$IE.WaitSettingOption’ or ‘$IE.WaitSettingScenario’ is specified for “wait_setting”, the setting in “wait_timeout” is not used.

The “wait_timeout” property of a node is output to the .wss7 file from WinActor only when ‘Use this “Property”’ is selected for “Timeout setting” in the “Property” pane of the node.

For "control," specify a parameter name for identify the target element. A parameter name including spaces, such as "frame index" for example, should be enclosed in quotes.

"<check>" attached to "Parameter name" of "control" is for specifying check box status of the Details tab displayed in the property of a node recorded in the IE mode.

<true> for checked and <false> for unchecked.

Any parameter names can be omitted.

The parameter names are as follows. The attributes of "control" for the IE mode recording is the same for the actions described below.

Table 8-15. Parameter names for Click

Parameter name	Description
tag	Specify an HTML tag name of the target element by entering a value or with a variable. A value should be a string.
'frame index'	Specify a serial number assigned to the target frame in a document by entering a value or with a variable. A value should be a number.
'tag index'	Specify a serial number assigned to the target element in a frame by entering a value or with a variable. A value should be a number.
ie_control_name	Specify a name attribute value of the target element by entering a value or with a variable. A value should be a string.
type	Specify a type attribute value of the target element by entering a value or with a variable. A value should be a string.
id	Specify an id attribute value of the target element by entering a value or with a variable. A value should be a string.
value	Specify a value attribute value of the target element by entering a value or with a variable. A value should be a string.

8.2.2 IE mode recording – Set Text

```
WinActor.SetTextIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = String or variable name,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,        // millisecons
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "value," specify a string to be set or a variable name that stores a string.

A constant of the "value" will be converted to a string even if it is a number.

When you do not set "value," specify the anonymous identifier '' (two single quotes) in "value." If you specify "" (two double quotes), it means that you specify an empty string.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.3 IE mode recording – Select Item in List

```
WinActor.SelectListIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = Expression or variable name,
    kind             = Item type,
    wait_setting     = Timeout option,
    wait_timeout     = Tmeout period,        // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

No value is returned.

For "kind," specify the item type $SelectListIE8.index or $SelectListIE8.text.

Table 8-16. Item types in the Select Item in List

Item type	Description
$SelectListIE8.index	Selects an item in a list by specifying a zero-based index
$SelectListIE8.text	Selects an item in a list by a string displayed in the list

For "value," specify an expression that results in a value for selecting a list box or a variable name that stores a value.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1IE mode recording – Click.”

8.2.4 IE mode recording – Get String

```
WinActor.GetTextIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = Variable name to receive the result,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,       // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

The result is returned to the variable specified in "value."

If WinActor.GetTextIE8 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign

statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.5 IE mode recording – Get Item in List

```
WinActor.GetListIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = Variable name to receive the result,
    kind             = Item type,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,        // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

The result is returned to the variable specified in "value."

If WinActor.GetListIE8 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For "kind," specify the item type $GetListIE8.index or $GetListIE8.text.

Table 8-17. Item types in Get Item in List

Item type	Description
$GetListIE8.index	Gets an index of a selected element in a list
$GetListIE8.text	Gets a name of a selected element in a list

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.6 IE mode recording – Get Check State

```
WinActor.GetCheckIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = Variable name to receive the result,
```

```
    wait_setting      = Timeout option,
    wait_timeout      = Timeout period,        // milliseconds
    capture           = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
)
```

True or false is returned to the variable specified in "value."

If WinActor.GetCheckIE8 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.7 IE mode recording – Get Enable/Disable State

```
WinActor.GetEnableIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    value            = Variable name to receive the result,
    wait_setting     = Timeout option,
    wait_timeout     = Timeout period,        // milliseconds
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

True or false is returned to the variable specified in "value."

If WinActor.GetEnableIE8 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.8 IE mode recording – Get Value in Table

```
WinActor.GetTableinfoIE8 Preamble
  (
    window_rule_ref      = Window_rule WinID name,
    control              = (Parameter name<check> = value, ...),
    get_tableinfo_mode   = Get mode,
    valuerow             = Specify a row number,
    valuecolumn          = Specify a column number,
```

```
    result          = Variable name to receive the result,
    file            = Filename string or variable name that stores a filename,
    wait_setting    = Timeout option,
    wait_timeout    = Timeout period,       // milliseconds
    capture         = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
)
```

For "get_tableinfo_mode," specify one of the following get modes.

Table 8-18. Get modes for Get Value in Table

Get mode	Description
$IEGetTableInfo.GetCell	Gets a value in a cell
$IEGetTableInfo.ExistCell	Checks the cell existence (true/false)
$IEGetTableInfo.GetRow	Gets the number of rows
$IEGetTableInfo.GetColumn	Gets the number of columns
$IEGetTableInfo.GetAll	Gets all values in a table

When the get mode is $IEGetTableInfo.GetCell or $IEGetTableInfo.ExistCell, specify a numeric value of a row number or a variable name that stores a row number in "valuerow," and specify a numeric value of a column number or a variable name that stores a column number in "valuecolumn."

If $IEGetTableInfo.GetAll is specified for the get mode, the result will be written to a file specified in "file" in CSV format.

If other than $IEGetTableInfo.GetAll is specified for the get mode, the result will be returned to a variable specified in "result."

If WinActor.GetTableinfoIE8 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.2.9 IE mode recording – Get All Items in List

```
WinActor.GetAllListIE8 Preamble
  (
    window_rule_ref  = Window_rule WinID name,
    control          = (Parameter name<check> = value, ...),
    file             = Variable name or filename,
```

```
    wait_setting      = Timeout option,
    wait_timeout      = Timeout period,        // milliseconds
    capture           = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
)
```

No value is returned.

For "file," specify a filename string to output the result or a variable name that stores a filename.

For “wait_setting” and “wait_timeout,” see the descriptions in “8.2.1 IE mode recording – Click.”

8.3 Action, User, Variable

8.3.1 Image Matching

```
WinActor.ImageMatch [_OriginalID_f63cb690ce1d = image ID number]
  (
    window_rule_ref  = Window_rule WinID name,
    targetrange      = (x = X-coordinate, y = Y-coordinate, width = width, height = height),
                                                  // Image search range
    rawtargetrange   =   (x = X-coordinate, y = Y-coordinate, width = width, height = height),
                                                  // Rectangular range for recording mouse cursor
    mousecoordinate = (enable = true / false, x = X-coordinate, y = Y-coordinate),
                                                  // Mouse action coordinates
    mouseaction      = Mouse action,
    scale            = Scale,                     // Scale
    selectshape      = Rectangle / Ellipse mode,  // Red-framed matching mode: Rectangle or Ellipse
    similarity       = Match ratio,               // 0-100 %
    timeout          = Timeout time,              // Milliseconds
    searchrange      = (enable = true / false , x = X-coordinate, width = width, y = Y-coordinate, height
= height, startpoint= origin, x_coordinate = x-coordinate value, y_coordinate = y-coordinate value),
    realtime         = true / false,              // true when loading matching images on execution
    realtimefile     = File path string or variable name,
                                                  // Filename or folder name when realtime = true
    useredframe      = true / false,              // true when using the red-framed reference image
                                                  // and the path-specified matching images
    pathspecified    = File / Folder,             // How to specify path-specified matching image files
    imagesplit       = true / false,              // true for subdivision matching
    value            = Variable name to receive the result,
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

The "_OriginalID_f63cb690ce1d" attribute of the preamble specifies the ID that identifies the image stored in WinActor, so do not change it.

You can add "comment" and "name" to the preamble.

The image file needs to be edited (changed or added) in WinActor.

The file will not be replaced even if you rename it in WSS.

Specifying " selectshape" is optional. If omitted, it is assumed to be the Rectangle mode.

When loading matching images on execution by specifying "realtime = true" and "useredframe = true," both the red-framed reference image and the path-specified matching images are used. Specifying "useredframe" is optional. The default value is false. Although "realtime = false" and "useredframe = true" can be specified simultaneously, only the red-framed reference image is used in that case.

Specifying "pathspecified" is optional. If omitted, it is assumed to be the File.

If the mouse action is "Matching only," the result will be returned to the variable specified in "value."

If WinActor.ImageMatch appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For "mouseaction," specify one of the following mouse actions.

Table 8-19. Mouse actions for Image Matching

Mouse action	Description
$ImageMatch.Check	Matching only
$ImageMatch.LeftClick	Left button click
$ImageMatch.RightClick	Right button click
$ImageMatch.LeftDouble	Left button double-click
$ImageMatch.RightDouble	Right button double-click
$ImageMatch.Move	Mouse cursor move
$ImageMatch.LeftTriple	Left button triple-click
$ImageMatch.RightTriple	Right button triple-click
$ImageMatch.LeftClickDrag	Holds left button and drags to the matched position
$ImageMatch.RightClickDrag	Holds right button and drags to the matched position

For "scale," specify one of the following scales.

Table 8-20. Scales for Image Matching

Scale	Description
$ImageMatch.Same	1x
$ImageMatch.Half	1/2
$ImageMatch.Quarter	1/4

For "selectshape," specify one of the following shape modes.

Table 8-21. Shape modes for Image Matching

Shape mode	Description
$ImageMatch.SelectShape_Rectangle	Rectangle
$ImageMatch.SelectShape_Ellipse	Ellipse

For "startpoint," specify one of the following origins.

Table 8-22. Origins for Image Matching

Origin	Description
$ImageMatch.StartPoint_LeftTop	Upper left
$ImageMatch.StartPoint_LeftBottom	Lower left
$ImageMatch.StartPoint_RightTop	Upper right
$ImageMatch.StartPoint_RightBottom	Lower right

For "x_coordinate" and "y_coordinate," specify one of the following coordinate values.

Table 8-23. Coordinate values for Image Matching

Coordinate value	Description
$ImageMatch.Coordinate_Direct	Coordinates are specified in pixels
$ImageMatch.Coordinate_Percent	Coordinates are specified in percentage

For "pathspecified," specify one of the following path-specified matching image files.

Table 8-24. Path-specified matching image files for Image Matching

Path-specified matching image files	Description
$ImageMatch.Path_File	File path

Path-specified matching image files	Description
$ImageMatch.Path_Folder	Folder path

8.3.2 Contour Matching

```
WinActor.OutlineMatch [_OriginalID_f63cb690ce1d = image ID number]
  (
    window_rule_ref  = Window_rule WinID name,
    targetrange      = (x = X-coordinate, y = Y-coordinate, width = width, height = height),
                                                  // Image search part
    mousecoordinate = (enable = true / false, x = X-coordinate, y = Y-coordinate),
                                                  // Mouse action coordinates
    mouseaction      = Mouse action,
    precision        = Precision,
    scale            = Scale,
    timeout          = Timeout time,              // Milliseconds
    searchrange      = (enable = true / false , x = X-coordinate, width = width, y = Y-coordinate, height
= height, startpoint= origin, x_coordinate = x-coordinate value, y_coordinate = y-coordinate value),
    realtime         = true / false,              // true when loading matching images on execution
    realtimefile     = File path string or variable name,
                                                  // Specify a filename when realtime = true
    useredframe      = true / false,               // true when using the red-framed reference image
                                                   // and the path-specified matching images
    imagesplit       = true / false,              // true for subdivision matching
    pathspecified    = File / Folder,             // How to specify path-specified matching image files
    value            = Variable name to receive the result,
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

The "_OriginalID_f63cb690ce1d" attribute of the preamble specifies the ID that identifies the image stored in WinActor, so do not change it.

You can add "comment" and "name" to the preamble.

The image file needs to be edited (changed or added) in WinActor.

The file will not be replaced even if you rename it in WSS.

When loading matching images on execution by specifying “realtime = true” and “useredframe = true,” both the red-framed reference image and the path-specified matching images are used. Specifying “useredframe” is optional. The default value is “false.” Although “realtime = false” and “useredframe = true” can be specified simultaneously, only the red-framed reference image is used in that case.

Specifying “pathspecified” is optional. If omitted, it is assumed to be the File.

If the mouse action is "Matching only," the result will be returned to the variable specified in "value."

If WinActor.OutlineMatch appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For "mouseaction," specify one of the following mouse actions.

Table 8-25. Mouse actions for Contour Matching

Mouse action	Description
$OutlineMatch.Check	Matching only
$OutlineMatch.LeftClick	Left button click
$OutlineMatch.RightClick	Right button click
$OutlineMatch.LeftDouble	Left button double-click
$OutlineMatch.RightDouble	Right button double-click
$OutlineMatch.Move	Mouse cursor move
$OutlineMatch.LeftTriple	Left button triple-click
$OutlineMatch.RightTriple	Right button triple-click
$OutlineMatch.LeftClickDrag	Holds left button and drags to the matched position
$OutlineMatch.RightClickDrag	Holds right button and drags to the matched position

For "precision," specify one of the following precisions.

Table 8-26. Precisions for Contour Matching

Precision	Description
$OutlineMatch.LowPrecision	Low (speed)
$OutlineMatch.MiddlePrecision	Middle (standard)
$OutlineMatch.HighPrecision	High (precision)

For "scale," specify one of the following scales.

Table 8-27. Scales for Contour Matching

Scale	Description
$OutlineMatch.Same	1x
$OutlineMatch.Half	1/2
$OutlineMatch.Quarter	1/4

For "startpoint," specify one of the following origins.

Table 8-28. Origins for Contour Matching

Origin	Description
$OutlineMatch.StartPoint_LeftTop	Upper left
$OutlineMatch.StartPoint_LeftBottom	Lower left
$OutlineMatch.StartPoint_RightTop	Upper right
$OutlineMatch.StartPoint_RightBottom	Lower right

For "x_coordinate" and "y_coordinate," specify one of the following coordinate values.

Table 8-29. Coordinate values for Contour Matching

Coordinate value	Description
$OutlineMatch.Coordinate_Direct	Coordinates are specified in pixels
$OutlineMatch.Coordinate_Percent	Coordinates are specified in percentage

For "pathspecified," specify one of the following path-specified matching image files.

Table 8-30. Path-specified matching image files for Image Matching

Path-specified matching image files	Description
$OutlineMatch.Path_File	File path
$OutlineMatch.Path_Folder	Folder path

8.3.3 OCR Matching

```
WinActor.OCRMatch [_OriginalID_f63cb690ce1d = image ID number]
  (
    window_rule_ref  = Window_rule WinID name,
    targetrange      = (x = X-coordinate, y = Y-coordinate, width = width, height = height),
                                                          // Image search part
    mousecoordinate = (enable = true / false, x = X-coordinate, y = Y-coordinate),
                                                          // Mouse action coordinates
    mouseaction      = Mouse action,
    timeout          = Timeout time,                      // Milliseconds
    searchrange      = (enable = true / false , x = X-coordinate, width = width, y = Y-coordinate, height
= height, startpoint= origin, x_coordinate = x-coordinate value, y_coordinate = y-coordinate value),
    ocrmatchingtext  = Matching string,
    value            = Variable name to receive the result,
```

```
    capture              = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
)
```

The "_OriginalID_f63cb690ce1d" attribute of the preamble specifies the ID that identifies the image stored in WinActor, so do not change it.

You can add "comment" and "name" to the preamble.

The image file needs to be edited (changed or added) in WinActor.

The file will not be replaced even if you rename it in WSS.

If the mouse action is "Matching only," the result will be returned to the variable specified in "value."

If WinActor.OCRMatch appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For "mouseaction," specify one of the following mouse actions.

Table 8-31. Mouse actions for OCR Matching

Mouse action	Description
$OCRMatch.Check	Matching only
$OCRMatch.LeftClick	Left button click
$OCRMatch.RightClick	Right button click
$OCRMatch.LeftDouble	Left button double-click
$OCRMatch.RightDouble	Right button double-click
$OCRMatch.Move	Mouse cursor move
$OCRMatch.LeftTriple	Left button triple-click
$OCRMatch.RightTriple	Right button triple-click
$OCRMatch.LeftClickDrag	Holds left button and drags to the matched position
$OCRMatch.RightClickDrag	Holds right button and drags to the matched position

For "startpoint," specify one of the following origins.

Table 8-32. Origins for OCR Matching

Origin	Description
$OCRMatch.StartPoint_LeftTop	Upper left
$OCRMatch.StartPoint_LeftBottom	Lower left
$OCRMatch.StartPoint_RightTop	Upper right

Origin	Description
$OCRMatch.StartPoint_RightBottom	Lower right

For "x_coordinate" and "y_coordinate," specify one of the following coordinate values.

Table 8-33. Coordinate values for OCR Matching

Coordinate value	Description
$OCRMatch.Coordinate_Direct	Coordinates are specified in pixels
$OCRMatch.Coordinate_Percent	Coordinates are specified in percentage

8.3.4 Wait for Window Status

```
WinActor.WindowStateWait Preamble
  (
    window_rule_ref  = Window_rule WinIDname,
    win_state        = Expected status,
    state            = Wait type,
    timeout          = Timeout time,
                       // Milliseconds   Effective when $Window.WaitFor is specified in "Wait type"
    value            = Variable name to receive the result,
    capture          = (imageid = image ID string, x = mouse coordinate X, y = mouse coordinate Y)
  )
```

For "win_state," specify one of the following expected status.

Table 8-34. Expected status for Wait for Window Status

Expected status	Description
$Window.Front	Window is at the front
$Window.Behind	Window is not at the front
$Window.Enable	Window is enabled
$Window.Disable	Window is disabled
$Window.Appear	Window is shown
$Window.Disappear	Window is hidden

For "state," specify one of the following wait types.

Table 8-35. Wait types for Wait for Window Status

Wait type	Description
$Window.WaitFor	Waits until timeout
$Window.CheckOnly	Gets status only

True or false is returned to the variable specified in "value."

If WinActor.WindowStateWait appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.5 Wait for Time

```
WinActor.TimerWait Preamble
  (
    mode        = Mode,
    timeout     = Timeout time,
                    // Milliseconds or variable name, when the mode is $TimerWait.Sleep
    wait_time   = Wait time,
                    // String or variable name, when the mode is $TimerWait.Until
    check_time  = Check time,
                    // String or variable name, when the mode is $TimerWait.Check
    check_value = Variable name to receive the check result,
    date_format = Date format,
    timezone    = Time zone
  )
```

For "mode," specify one of the following modes:

Table 8-36. Modes for Wait for Time

Mode	Description
$TimerWait.Sleep	Waits for the specified time
$TimerWait.Until	Waits until the specified time
$TimerWait.Check	Checks if the specified time comes

For "date_format," specify one of the following or the date format string allowed by WinActor.

Table 8-37. Date formats for Wait for Time

Date format	Description
$TimerWait.ScenarioInfoDateFormat	Specifies a format on the Scenario information property
$TimerWait.OptionInfoDateFormat	Specifies a format on the Option dialog

For "timezone," specify one of the following or the time zone string allowed by WinActor.

Table 8-38. Time zones for Wait for Time

Time zone	Description
$TimerWait.ScenarioInfoTimeZone	Specifies a format on the Scenario information property
$TimerWait.OptionInfoTimeZone	Specifies a format on the Option dialog
$TimerWait.DefaultTimeZone	OS default

If "mode" is to "check if the specified time comes," true or false will be returned to the variable specified in "check_value."

For details, see "Time format" under the "Wait for Time" node in "WinActor Operation Manual."

If WinActor.TimerWait appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the check result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the check result is omitted.

8.3.6 Send Text

WinActor.SendText *Preamble*
(
window_rule_ref = Window_rule *WinIDname*,
control = (instance<*check*> = *expression*, text<*check*> = *expression*, position<*check*> = *position*),
value = *String or variable name*,
sendcr = true or false,
verify = true or false,
capture = (imageid = *image ID string*, x = *mouse coordinate X*, y = *mouse coordinate Y*)
)

No value is returned.

For "value," specify a string to be sent or a variable name that stores the sent contents. When you do not set "value," specify the anonymous identifier ''.

A constant of the "value" attribute will be converted to a string even if it is a number.

For "sendcr" and "verify," specify true or false. If omitted, it is assumed that false is specified.

A warning will be issued when a value other than true or false is specified, and the process will continue assuming that false is specified.

The meanings of "sendcr" and "verify" are as follows.

Table 8-39. "sendcr" and "verify"

Action	Description
sendcr	Sends the return key
verify	Verifies a sent result (Pauses in case of verification error)

8.3.7 Execute Command

```
WinActor.Launcher Preamble
  (
    command      = Command name or variable name that stores the command name,
    option       = Option string or variable name that stores the option string,
    execute_mode = Execution mode,
    set_value    = Variable name to receive the result
  )
```

For "execute_mode," specify one of the following execution modes.

Table 8-40. Execution modes for Execute Command

Execution mode	Description
$Launcher.Single	Asynchronous execution (single instance)
$Launcher.Multi	Asynchronous execution
$Launcher.WaitForEnd	Synchronous execution (receives result)

If "execute_mode = $Launcher.WaitForEnd" is specified, the result value will be returned to the variable specified for "set_value."

If WinActor.Launcher appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.8 Run Script

```
WinActor.Script Preamble
  (
      window_rule_ref      = Window_rule WinID name,
      environment_type     = Variable sharing type,
      library_provider     = Library provider (string),
      library_id           = Library ID (string),
      library_version      = Library version (string),
      library_original_name = Library name (string),
      edit_lock            = true or false,

      Script parameter description = value or variable name depending on the description,
          // Arrange this line by the number of parameters

      /*** Note ***/
      note = @"""
Annotation description
""",
      /*** Note End ***/
      /*** Script ***/
      script = @"""
Script description
"""
      /*** Script End ***/
  )
```

Annotation and script should be written between the line that ends with @""" and the line that starts with """.

The script parameter description should be enclosed in '' (two single quotes) and used as an identifier.

"Parameter name" is a name for exchanging a value with WinActor in the script description.

Table 8-41. Script parameter descriptions for Run Script

Script parameter description	Description
'!Parameter_name!'	Variable name or value
'!Parameter_name|Item1,Item2,Item3...!'	Pull-down menu
'$Parameter_name$'	Variable name
'@Parameter_name@'	WinID name
'!Parameter_name|FILE|!'	File selection dialog
'!Parameter_name|FILE:EXCEL|!'	File selection dialog (Excel)

Script parameter description	Description
'!Parameter_name|FILE:ZIP|!'	File selection dialog (zip)
'!Parameter_name|FILE:CSV|!'	File selection dialog (csv)
'!Parameter_name|FILE:IMG|!'	File selection dialog (image)

For "environment_type," specify one of the following variable sharing types:

Table 8-42. Variable sharing types for Run Script

Variable sharing type	Description
$Script.EnvIndependent	Sets variables for each script
$Script.EnvShared	Shares variables with other scripts

The Run Script node of each sample library has version information parameters (library_provider, library_id, library_version, library_original_name). These parameters cannot be changed in WSS.

If "edit_lock" is omitted, it will be assumed that false is specified.

Regarding the Run Script node with its script locked (edit_lock) in WinActor, its script description will not be displayed in WSS and cannot be changed.

No value is returned.

Example

```
WinActor.Script [name = "Countdown", comment = ""]
(
    window_rule_ref = "",
    environment_type = $Script.EnvIndependent,
    library_provider = "NTT Advanced Technology Corporation ",
    library_id = "AT05001L",
    library_version = "1.0.0",
    library_original_name = "Countdown",
    edit_lock = false,
    '$Counter$' = a,
    /*** Note ***/
    note = @"""
Counts down a number of the specified variable.

Counter: Specify a variable name of the countdown target.
""",
    /*** Note End ***/
```

```
      /*** Script ***/
      script = @"""
c = GetUMSVariable( $Counter$ )
ci = int(c)
ci = ci - 1
SetUMSVariable $Counter$, ci
"""
      /*** Script End ***/
);
```

8.3.9 Run Python

```
WinActor.PythonScript Preamble
  (
      window_rule_ref      = Window_rule WinID name,
      environment_type     = Variable sharing type,
      library_provider     = Library provider (string),
      library_id           = Library ID (string),
      library_version      = Library version (string),
      library_original_name = Library name (string),
      edit_lock            = true or false,

      Script parameter description = value or variable name depending on the description,
          // Arrange this line by the number of parameters

      /*** Note ***/
      note = @"""
Annotation description
""",
      /*** Note End ***/
      /*** Script ***/
      script = @""""
PythonScript description
""""
      /*** Script End ***/
  )
```

Annotation should be written between the line that ends with @""" , which includes three double quotes, and the line that starts with """, which is three double quotes.

Python script should be written between the line that ends with @"""" , which includes four double quotes, and the line that starts with """", which is four double quotes, to avoid interfering with the other part of the Python script.

The script parameter description should be enclosed in ' ' (single quotes) and used as an identifier.

"Parameter name" is a name for exchanging a value with WinActor in the Python script description. The descriptions are the same as listed in the “Table 8-41. Script parameter descriptions for Run Script.”

Table 8-43. Script parameter descriptions for Run Python

Script parameter description	Description
'!Parameter_name!'	Variable name or value
'!Parameter_name|Item1,Item2,Item3...!'	Pull-down menu
'$Parameter_name$'	Variable name
'@Parameter_name@'	WinID name
'!Parameter_name|FILE|!'	File selection dialog
'!Parameter_name|FILE:EXCEL|!'	File selection dialog (Excel)
'!Parameter_name|FILE:ZIP|!'	File selection dialog (zip)
'!Parameter_name|FILE:CSV|!'	File selection dialog (csv)
'!Parameter_name|FILE:IMG|!'	File selection dialog (image)

For "environment_type," specify one of the following variable sharing types.

Table 8-44. Variable sharing types for Run Python

Variable sharing type	Description
$Script.EnvIndependent	Sets variables for each Python script
$Script.EnvShared	Shares variables with other Python scripts

The Run Python node of each sample library has version information parameters (library_provider, library_id, library_version, library_original_name). These parameters cannot be changed in WSS.

If "edit_lock" is omitted, it will be assumed that false is specified.

Regarding the Run Python node with its script locked (edit_lock) in WinActor, its script description will not be displayed in WSS and cannot be changed.

No value is returned.

Example

```
WinActor.PythonScript [name = "Countdown", comment = ""]
```

```
(
    window_rule_ref = "",
    environment_type = $Script.EnvIndependent,
    library_provider = " ",
    library_id = "",
    library_version = "",
    library_original_name = "",
    edit_lock = false,
    '!String1!' = "first",
    '!String2!' = "second",
    '$ConcatenatedResult$' = result_py,        /*** Note ***/
    note = @"""
Concatenates two strings
""",
    /*** Note End ***/
    /*** Script ***/
    script = @"""
result = "".join([!String1!,    !String2!])
vm_result = $ConcatenatedResult$

winactor.set_variable(vm_result, result)
"""
    /*** Script End ***/
);
```

8.3.10 Excel Operation

```
WinActor.Excel Preamble
  (
    operation      = Excel operation,
    file_path      = Excel file path or variable name,
                                                  // Required for any operation
    sheet          = Sheet name or variable name,
                                                  // Required when the operation is to set or get a value
    cell           = Cell position or variable name,
                                                  // Required when the operation is to set or get a value
    source_value   = Value to be set or variable name,
                                                  // Required when the operation is to set a value
    target_value   = Variable name to receive the result,
                                                  // Required when the operation is to get a value
    macro          = Macro name or variable name
                                                  // Required when the operation is to run a macro
  )
```

For "operation," specify one of the following Excel operations.

Table 8-45. Excel operations

Excel operation	Description
$Excel.GetValue	Gets a value
$Excel.SetValue	Sets a value
$Excel.RunMacro	Runs a macro

"Or variable name" means the name of a variable that stores necessary information.

If the operation is to get a value, the result will be returned to the variable specified in "target_variable."

If WinActor.Excel appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.11 Clipboard

```
WinActor.Clipboard Preamble
  (
     mode      = Operation,
     set_value = Expression,                    // Required when the operation is $ClipBoard.Set
     get_value = Variable name to receive the result
  )
```

For "mode," specify one of the following operations.

Table 8-46. Clipboard operations

Clipboard operation	Description
$ClipBoard.Set	Sets a value to the clipboard
$ClipBoard.Get	Gets a value from the clipboard

When setting a value to the clipboard, "get_value" can be omitted.

When getting a value from the clipboard, the value is returned to the variable specified in "get_value," and "set_value" can be omitted.

If WinActor.Clipboard appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

The following WinActor.SetToClipboard and WinActor.GetFromClipboard can be used with less arguments and are easier to understand.

8.3.12 Set To Clipboard

```
WinActor.SetToClipboard Preamble
  (
      String or variable name that stores contents to be set in the clipboard
  )
```

No value is returned.

8.3.13 Get From Clipboard

```
WinActor.GetFromClipboard Preamble
  (
    get_value = Variable name to receive the result
  )
```

The result value is returned to the variable specified in "get_value."

If WinActor.GetFromClipboard appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional, but () are required. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the result is omitted.

Example

```
cv = WinActor.GetFromClipboard [ name = "get from clipboard" ] ();
```

8.3.14 Waiting Dialog

```
WinActor.WaitBox Preamble
  (
    mode    = Mode,
    message = Message string or variable name that stores a message
  )
```

For "mode," specify one of the following modes:

Table 8-47. Modes for Waiting Dialog

Mode	Description
$WaitBox.Confirm	Confirmation dialog (displays OK button only)
$WaitBox.Query	Inquiry dialog (displays Continue and Stop buttons)

No value is returned.

8.3.15 Input Dialog

```
WinActor.InputBox Preamble
  (
    message = Message string or variable name that stores a message,
    value   = Variable name to receive the result
  )
```

The result is returned to the variable specified in "value."

If WinActor.InputBox appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

A constant of the "message" attribute will be converted to a string even if it is a number.

8.3.16 Selection Dialog

```
WinActor.SelectBox Preamble
  (
    message = Message string or variable name that stores a message,
    items   = Option list,
    value   = Variable name to receive the result
  )
```

Specify "option list" of "items" by enclosing option strings in parentheses.

You can use a string or a constant that stores a string.

Numbers are treated as strings.

Example:

```
items = ("red", "blue", "white"),
items = (1, 2, 3),
```

The result is returned to the variable specified in "value."

If WinActor.SelectBox appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.17 Sound (Buzzer)

```
WinActor.Beep Preamble ()
```

This is to make a buzzer sound.

There is no argument.

No value is returned.

8.3.18 Sound (WAVE file)

```
WinActor.Speaker    [name = "name", comment = "comment", _OriginalID_f63cb690ce1d = 268]
  (
    selectFile  = WAVE filename,
    wait        = true / false          // true when waiting until the playback ends
  )
```

The "_OriginalID_f63cb690ce1d" attribute of the preamble specifies the ID that identifies the WAVE file stored in WinActor, so do not change it.

You can add "comment" and "name" to the preamble.

The WAVE file needs to be edited (changed or added) in WinActor.

The file will not be replaced even if you rename it in WSS.

No value is returned.

8.3.19 Set Variable Value

```
Variable name to receive the result = Expression Preamble ;
```

Set Variable Value is written by using an assignment statement.

You can set "comment" and "name" in the preamble.

When writing as an action:

```
WinActor.SetVariable Preamble (
  val     = Constant expression,
  value = Variable name to receive the result
```

```
)
```

The result of a constant expression is returned to the variable specified in "value."

If WinActor.SetVariable appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

When written by using an assignment statement, "val = " can be omitted.

Example: All of the following have the same result.

```
WinActor.SetVariable(val = 100, value = height);
height = WinActor.SetVariable(val = 100);
height = WinActor.SetVariable(100);
height = 100;
```

8.3.20 Copy Variable Value

```
Variable name to receive the result = Expression Preamble ;
```

Copy Variable Value is written by using an assignment statement.

You can set "comment" and "name" in the preamble.

When writing as an action:

```
WinActor.CopyVariable Preamble
(
    from = Source variable name,
    to   = Destination variable name,
)
```

The contents of the source variable is copied to the variable specified in "to."

If WinActor.CopyVariable appears in an assignment or expression, the result will not be stored in the variable specified in "to." Specifying "to" is optional.

When written by using an assignment statement, "from = " can be omitted.

Example: All of the following have the same result.

```
WinActor.CopyVariable(from = height, to = width);
width = WinActor.CopyVariable(from = height);
width = WinActor.CopyVariable(height);
width = height;
```

8.3.21 Get Date and Time

```
WinActor.GetDateTime Preamble
  (
    format      = Format type,
    date_format = Date format,
    timezone    = Time zone,
    value       = Variable name to receive the result
  )
```

For "format," specify one of the following format types.

Table 8-48. Format types for Get Date and Time

Format type	Description
$GetDateTime.DateTime	Date and time
$GetDateTime.Date	Only date
$GetDateTime.Time	Only time

For "date_format," specify one of the following or the date format string allowed by WinActor.

Table 8-49. Date format for Get Date and Time

Date format	Description
$GetDateTime.ScenarioInfoDateFormat	Specifies a format on the Scenario information property (This date format is applied when "date_format" is omitted)
$GetDateTime.OptionInfoDateFormat	Specifies a format on the Option dialog

For "timezone," specify one of the following or the time zone string allowed by WinActor.

Table 8-50. Time zone for Get Date and Time

Time zone	Description
$GetDateTime.ScenarioInfoTimeZone	Specifies a time zone on the Scenario information property (This time zone is applied when "timezone" is omitted)
$GetDateTime.OptionInfoTimeZone	Specifies a time zone on the Option dialog
$GetDateTime.DefaultTimeZone	OS default

"date_format" and "timezone" can be omitted.

The result is returned to the variable specified in "value."

If WinActor.GetDateTime appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.22 Get Username

```
WinActor.GetUserName Preamble
  (
    value = Variable name to receive the result
  )
```

The result is returned to the variable specified in "value."

If WinActor.GetUserName appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional, but () are required. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the result is omitted.

Example:

```
uname = WinActor.GetUserName [Comment = "Get username"] ();
```

8.3.23 Four Arithmetic Operations

Four Arithmetic Operations is written by using an expression and assignment statement.

When writing as an action:

```
WinActor.Calculate Preamble (
  operator  = Operation,
  left      = Left operand of the binary operation,
  right     = Right operand of the binary operation,
  value     = Variable name to receive the result
)
```

For "operator," specify one of the following operations.

Table 8-51. Four Arithmetic Operations

Operation
$Calculate.Plus
$Calculate.Minus
$Calculate.Mul

Operation
$Calculate.Div

Numbers, variable names, and expressions can be written in the operands of the binary operation.

When an expression is written, the compiler automatically generates a node for the operation of the expression.

The operation result is returned to the variable specified in "value."

If WinActor.Calculate appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.3.24 Count Up

Count Up is written by using an expression and assignment statement.

When writing as an action:

```
WinActor.CountUP Preamble (
  value = Variable name,
  add   = Incremental constant expression
)
```

No value is returned.

The incremental constant is added to the variable specified in "value."

If a negative value is specified for the constant, a warning will be issued in WinActor.

8.3.25 Full/Half-Width Conversion

```
WinActor.TextConvert Preamble
  (
    operation       = Operation,
    target_variable = Variable name that stores a conversion target string
  )
```

For "operation," specify one of the following operations.

Table 8-52. Operations for Full/Half-Width Conversion

Operation	Description
$FullHalfWidth.ToFullWidth	Converts to full-width characters
$FullHalfWidth.ToHalfWidth	Converts to half-width characters

No value is returned.

8.3.26 Watch Events

WinActor.EventsWatch *Preamble* ()

This begins watching events set in Events and calls the corresponding action when an event trigger is detected.

There is no argument.

8.3.27 Register EventWatcher

WinActor.EventAdd *Preamble* (Events *EventWatcher name*)

This registers an event in Events as a EventWathcer.

Specify EventWatcher name to register as an argument.

8.3.28 Cancel EventWatcher

WinActor.EventRemove *Preamble* (Events *EventWatcher name*)

This cancels an EventWatcher.

Specify EventWatcher name to cancel as an argument.

8.3.29 Ignore Events

WinActor.EventsIgnore *Preamble* ()

This terminates watching events and begins ignoring event triggers.

There is no argument.

8.4 WinActor Mail, HTTP, JSON

8.4.1 Mail Reception Settings

```
WinActor.MailReceiveSet Preamble
    (
        mail_rule_info      = Mail reception conditions,
        host_name           = String or variable,     // Hostname of the incoming mail server
        user                = String or variable,     // Username to connect to the incoming mail server
        pass                = String or variable,     // Password to connect to the incoming mail server
        auth_type           = Authentication type,    // See below
        port                = Port number,            // Integer value 110, etc.
        conn_time           = Connection timeout,     // Milliseconds 10000, etc. (10 seconds)
        cmd_time            = Reception timeout,      // Milliseconds 10000, etc. (10 seconds)
        mail_input          = String or variable,     // Mail folder
        is_del_mail         = true / false,           // Whether to delete received mails from server
        attach_save         = true / false,           // Whether to save attached files
        extention_not_save  = true / false,   // Space-separated, valid when extention_not_save is true
        except_extension    = "*.exe *.bat *.vbs *.msi *.jar",
                                              // Specify attached file extensions not to save
        security_type       = Secure connection type
                                              // See below
    );
```

This is to configure the mail reception settings.

No value is returned.

For "auth_type," specify one of the following authentication types.

Table 8-53. Authentication types for Set Mail Reception

Authentication type	Description
$MailAuth.UserPass	USER/PASS
$MailAuth.APOP	APOP

For "security_type," specify one of the following secure connection types.

Table 8-54. Secure connection types for Set Mail Reception

Secure connection type	Description
$MailSecurity.No	None
$MailSecurity.TLS_SSL	POP3S
$MailSecurity.StartTLS	StartTLS

- Mail reception conditions "mail_rule_info"

Write the conditions for receiving mails. You can write multiple conditions by separating them with commas. Only mails that meet all the conditions will be received.

The syntax of the condition is as follows:

(item = *Item*, cond = *Condition*, value = *Value*) *or* (item = *Item*, cond = *Condition*, var = *Variable name*)

For "item ," specify one of the following items.

Table 8-55. Items for mail reception conditions

Item	Description
$MailRule.To	Recipient
$MailRule.From	Sender
$MailRule.Subject	Subject

For "cond," specify one of the following conditions:

Table 8-56. Mail reception conditions

Condition	Description
$MailRule.Include	Include
$MailRule.AtFirst	Start with
$MailRule.AtLast	End with
$MailRule.Equal	Match
$MailRule.Regex	Regular expression

Example: Receiving mails with "example.com" included in the sender and with "report" included in the subject.

```
mail_rule_info =
    (
        (item = $MailRule.From, cond = $MailRule.Include, value = "example.com"),
        (item = $MailRule.Subject, cond = $MailRule.Include, value = "report")
    ),
```

8.4.2 Receive Mail

WinActor.MailReceive *Preamble*

```
(
    get_method        = Reception type,
    no_receiver_mail  = Operation in case of no mails,
    get_mail_num      = Variable name to receive the result
)
```

This is to receive mails.

For "get_method," specify one of the following reception types.

Table 8-57. Reception types for Receive Mail

Reception type	Description
$MailReceive.OneByOne	Receives one by one
$MailReceive.GetAll	Receives all mails
$MailReceive.NumOnly	Receives the number of mails

For "no_receiver_mail," specify one of the following operations.

Table 8-58. Operations in case of no mails for Receive Mail

Operation in case of no mails	Description
$MailReceive.Wait	Waits until mails can be received
$MailReceive.Error	Raises an error
$MailReceive.MailNum	Returns the number of mails (zero)

The number of received mails is returned to the variable specified in "get_mail_num."

If WinActor.MailReceive appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.4.3 Select Mail

```
WinActor.MailSelect Preamble
    (
        select_type             = Mail selection type,
        line_num                = Variable name to store the line number of the selected mail,
        not_mail_error_return = true or false
    )
```

This is to select a received mail.

For "select_type," specify one of the following mail selection types.

Table 8-59. Mail selection types for Select Mail

Mail selection type	Description
$MailSelect.Top	Selects the first mail
$MailSelect.NoProcessedTop	Selects the first unprocessed mail
$MailSelect.ProcessedTop	Selects the first processed mail
$MailSelect.Next	Selects the next mail
$MailSelect.NextNoProcessed	Selects the next unprocessed mail
$MailSelect.NextProcessed	Selects the next processed mail

The line number of the selected mail is returned to the variable specified in "line_num."

If WinActor.MailSelect appears in an assignment or expression, the result will not be stored in the variable. Specifying the "line_num" variable is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the line number is omitted.

8.4.4 Change Mail State

```
WinActor.MailStatusChg Preamble
  (
    $MailStatusChg.Processed or $MailStatusChg.NoProcessed
  )
```

Specify $MailStatusChg.Processed or $MailStatusChg.NoProcessed in the argument.

It means to change the state to "Processed" or "Unprocessed," respectively.

No value is returned.

8.4.5 Synchronize Mail Folder

```
WinActor.MailSync Preamble ()
```

This is to synchronize the Mail pane with the mail folder.

See "WinActor Mail Reception Scenario Creation Manual" for details.

There is no argument.

No value is returned.

8.4.6 Delete Processed Mail

```
WinActor.MailRemoveProcessed Preamble
 (
     deleted_mail_num = Variable name to receive the result
 )
```

The number of deleted mails is returned to the variable specified in "deleted_mail_num."

If WinActor.MailRemoveProcessed appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.4.7 Delete Mail

```
WinActor.MailRemove Preamble ()
```

This is to delete a selected mail.

There is no argument.

No value is returned.

8.4.8 Copy Mail Information

```
WinActor.MailCopyClip Preamble
   ( Argument )
```

This is to copy information of a selected mail to the clipboard.

No value is returned.

Specify one of the following in the argument.

Table 8-60. Arguments for Copy Mail Information

Argument	Description
$MailCopyClip.UniqueID	Unique ID
$MailCopyClip.FolderName	Folder name
$MailCopyClip.Status	State (unprocessed/processed)
$MailCopyClip.SendDate	Sent date
$MailCopyClip.From	Sender

Argument	Description
$MailCopyClip.Subject	Subject
$MailCopyClip.Body	Body
$MailCopyClip.NumberOfAttached	Number of attached files

8.4.9 Get Attached Filename

```
WinActor.MailAttachName Preamble
    (
        attach_file_number = Attached file number,
        attach_file_name   = Variable name to receive the filename
    );
```

The attached filename is returned to the variable specified in "attach_file_name."

If WinActor.MailAttachName appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the filename is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the filename is omitted.

8.4.10 Get Mail Information

```
WinActor.MailGetInfo Preamble
    (
        uid        = Variable name to receive the unique ID,
        dir        = Variable name to receive the folder name,
        stat       = Variable name to receive the mail state (unprocessed/processed),
        send_date  = Variable name to receive the sent date,
        from       = Variable name to receive the sender,
        to         = Variable name to receive the recipient,
        cc         = Variable name to receive the CC recipient,
        subject    = Variable name to receive the subject,
        message    = Variable name to receive the body,
        attachment = Variable name to receive the number of attached files
    );
```

This is to get information of a received mail.

For items that do not require information, specify the anonymous identifier '' (two single quotes) as a variable name or omit the item.

The information is returned to the specified variables. The result of the whole operation is not returned.

8.4.11 Import Mail Reception Settings

```
WinActor.MailReceiveImport Preamble
  (
     File path or variable name that stores a file path
  )
```

This is to import the mail reception settings.

No value is returned.

8.4.12 Gmail Reception Settings

```
WinActor.GmailReceiveSet Preamble
    (
        mail_rule_info      = Mail reception conditions,
        conn_time           = Connection timeout,   // Milliseconds 10000, etc. (10 seconds)
        cmd_time            = Reception timeout,    // Milliseconds 10000, etc. (10 seconds)
        mail_input          = String or variable,   // Mail folder
        attach_save         = true / false,         // Whether to save attached files
        extention_not_save  = true / false,   // Space-separated, valid when extention_not_save is true
        except_extension    = "*.exe *.bat *.vbs *.msi *.jar",
                                                // Specify attached file extensions not to save
    );
```

This is to configure Gmail reception settings.

The minimum value for the connection timeout and the reception timeout is 100, and the maximum value is 3,600,000 milliseconds.

No value is returned.

- Mail reception conditions “mail_rule_info”

Write the conditions for receiving mails. You can write multiple conditions by separating them with commas. Only mails that meet all the conditions will be received.

The syntax of the condition is as follows:

```
(item = Item, cond = Condition, value = Value)
or
(item = Item, cond = Condition, var = Variable name)
```

For "item," specify one of the following items.

Table 8-61 Items for mail reception conditions

Item	Description
$GmailRule.To	Recipient
$GmailRule.From	Sender
$GmailRule.Subject	Subject

For "cond," specify one of the following conditions:

Table 8-62 Mail reception conditions

Condition	Description
$GmailRule.Include	Include
$GmailRule.AtFirst	Start with
$GmailRule.AtLast	End with
$GmailRule.Equal	Match
$GmailRule.Regex	Regular expression

Example: Receiving mails with a string "example.com" included in the sender and with a string "report" included in the subject.

```
mail_rule_info =
    (
        (item = $MailRule.From, cond = $MailRule.Include, value = "example.com"),
        (item = $MailRule.Subject, cond = $MailRule.Include, value = "report")
    ),
```

8.4.13 Receive Gmail

```
WinActor.GmailReceive Preamble
    (
        get_method        = Reception type,
        no_receiver_mail  = Operation in case of no mails,
        get_mail_num      = Variable name to receive the result
    )
```

This is to receive mails via Gmail.

For "get_method," specify one of the following reception types.

Table 8-63 Reception types for Receive Gmail

Reception type	Description
$GmailReceive.OneByOne	Receives one by one
$GmailReceive.GetAll	Receives all mails
$GmailReceive.NumOnly	Receives the number of mails

For "no_receiver_mail," specify one of the following operations.

Table 8-64 Operations in case of no mails for Receive Gmail

Operation in case of no mails	Description
$GmailReceive.Wait	Waits until mails can be received
$GmailReceive.Error	Raises an error
$GmailReceive.MailNum	Returns the number of mails (zero)

The number of received mails is returned to the variable specified in "get_mail_num."

If WinActor.GmailReceive appears in an assignment or an expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

8.4.14 Gmail Send Settings

```
WinActor.GmailSendSet Preamble
    (
        conn_time          = Connection timeout,   // Milliseconds 10000, etc. (10 seconds)
        cmd_time           = Reception timeout,    // Milliseconds 10000, etc. (10 seconds)
    );
```

This is to configure the Gmail send settings.

The minimum value for the connection and the reception timeout is 100, and the maximum value is 3,600,000 milliseconds.

No value is returned.

8.4.15 Send Gmail

```
WinActor.GmailSend Preamble
    (
        recipient_name          = String or variable,  // Recipient's name
        recipient_address       = String or variable,  // Recipient's mail address
```

```
        subject                 = String or variable,  // Subject of the mail
        body                    = String or variable,  // Mail body
    );
```

This is to send a mail via Gmail.

No value is returned.

8.4.16 HTTP

```
WinActor.Http Preamble
  (
      method          = Method,
      url             = Server address URL string or variable name,
      server_timeout  = Connection timeout,                          // In milliseconds
      res_timeout     = Response timeout,                            // In milliseconds

      req_header      = (Key = value or variable name, ...),
      req_body_file   = File path string to store the request details or variable name,
      req_body        = (Key<Type> = value or variable name, ...),
      req_use_file    = true / false,   // When true is specified, req_body_file is adopted.

      res_header      = (Key = destination variable name, ...),
      res_body_file   = Destination file path string or variable name,
      res_body        = (Key = destination variable name, ...)
      res_use_file    = true / false,   // When true is specified, res_body_file is adopted.
      status_code     = Variable name to receive the status code
  )
```

For "method," specify one of the following:

Table 8-65. Methods for HTTP

Method
$HTTP.Get
$HTTP.Put
$HTTP.Post
$HTTP.Delete
$HTTP.Patch

Specify either "req_body_file" or "req_body." If both are specified, "req_body_file" will be taken when “req_use_file” is true, and .”req_body” when false. If both are specified and “req_use_file” is omitted, “req_body_file” will be taken.

Specify either "res_body_file" or "res_body." If both are specified, "res_body_file" will be taken when “res_use_file” is true, and “res_body” when false.. If both are specified and “res_use_file” is omitted, “res_body_file” will be taken.

If $HTTP.Get is specified in the method, "req_body_file" and "req_body" will be ignored.

The valid arguments (○) and ignored arguments (×) for each method are as follows.

Table 8-66. Methods and arguments for HTTP

Argument	Method					Description
	Get	Put	Post	Delete	Patch	
req_header	○	○	○	○	○	Key = value or variable name
req_body_file	×	○	○	○	○	File path string to store the request details or variable name
req_body	×	○	○	○	○	Key<type> = value or variable name Converted to JSON object and sent
res_header	○	○	○	○	○	Key = destination variable name
res_body_file	○	○	○	○	○	Destination file path string or variable name
res_body	○	○	○	○	○	Key = destination variable name The result will be in JSON format.

For "Type," specify one of the following:

Table 8-67. Types for HTTP

Type	Description
integer	Integer
float	Decimal
string	String
object	Object
array	Array
boolean	Boolean
null	Null

The status code is returned to the variable specified in "status_code."

If WinActor.Http appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the status code is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For details and notes, see "HTTP" in "WinActor User Library Sample Manual."

8.4.17 HTTP (Advanced)

```
WinActor.Http2 Preamble
  (
    method          = Method,
    url             = Server address URL string or variable name,
    proxy_mode      = true / false,   // Whether to use the proxy server set in option window
    response_timeout = Response timeout,   // in milliseconds
    auth            = true / false,   // \whether to use the BASIC authentication
    auth_user       = User name string or variable name,
    auth_pass       = Password string or variable name,
    req_header      = Request header in JSON-format string or variable name,
    req_header_file = Path name for the file that stores the request header or variable name,
    req_param_multipart = true / false,     //   Whether to send parameters in multipart
    req_param       = Parameters in JSON-format string or variable name,
    req_param_file = Path name for the file that stores the parameters or variable name,
    req_body        = Request body string,
    req_body_file = Path name for the file that stores the request body,
    req_cookie      = Request header cookie in JSON-format string or variable name,
    req_cookie_file = Path name for the file that stores the request header cookie or variable name,
    req_do_not_base64_attrs = true / false,
          // Whether not to base64 encode the 'Filename' and 'name' attributes when uploading a file
    req_fileupload_file = Path name for the file that stores the upload files setting or variable name,
    fileupload_info = Upload files list,
    res_header      = Variable name to receive response header,
    res_header_file = Path name for the file to receive response header or variable name,
    res_header_filetype = Response header format,
    res_body        = Variable name to receive response body,
    res_body_file = Path name for the file to receive response body or variable name,
    res_multipartdata_split = true / false, // Whether to split multipart data
    res_cookie      = Variable name to receive response cookie,
    res_cookie_file = Path name for the file to receive response cookie or variable name,
    res_cookie_filetype = Response cookie format,
    http_version = Variable name to receive the HTTP version,
    reason_phrase = Variable name to receive the reason phrase,
    status_code = Variable name to receive the status code
  )
```

Since many parameters exist, not selected or omitted parameters in .wss7 file are also output as comments on the file.

Specify one of the followings for “Method.”

Table 8-68. Method of HTTP2

Method
$HTTP.Get
$HTTP.Put
$HTTP.Post
$HTTP.Delete
$HTTP.Patch
$HTTP.Head

The “response_timeout” is optional. The default value is 10000, which means 10 seconds.

When ‘false’ is set for “auth,” both “auth_user” and “auth_password” are optional.

For following items in each row, the parameter on either left or right side should be specified. If the parameters on both sides are specified, the right side one is adopted.
When the parameters on both sides in the row are not necessary for the request, they are optional.

Table 8-69. Req arguments of HTTP2

From variable or value	From file
req_header	req_header_file
req_param	req_param_file
req_cookie	req_cookie_file
req_body	req_body_file

For following items in each row, the parameter on either left or right side should be specified to store the corresponding element in the response. If the parameters on both sides are specified, the right side one is adopted.

When the corresponding element is not necessary to store, parameters on both sides are optional.

Table 8-70. Res arguments of HTTP2

To variable	To file
res_header	res_header_file
res_cookie	res_cookie_file
res_body	res_body_file

For “res_header_filetype” and “res_cookie_filetype,” specify $HTTP2.RawFormat or $HTTP2.JSONFormat.

These parameters are optional. The default value for them is $HTTP2.RawFormat.

Table 8-71. Response format of HTTP2

Response format	Description
$HTTP2.RawFormat	Separated by new lines
$HTTP2.JSONFormat	JSON format

For “req_do_not_base64_attrs,” specify true or false.

Specify true to not encode the “Filename” and “name” attributes in base64, or false to encode them in base64, when sending them with the file to upload. The “req_do_not_base64_attrs” is optional. The default value is false on the scenario created with WinActor 7.6.0 or earlier, and true on the scenario created with WinActor 7.6.1 or later.

For the upload files, specify either “fileupload_info” or “req_fileupload_file.”

If both of them are specified, “req_fileupload_file” is adopted.

The syntax of “fileupload_info” is as follows. List triplets of a filename, a form name and a content type for all the files to upload.

```
fileupload_info = (
      (filename = Path of a file or variable name,
       name = String or variable name,
       content_type = String or variable name),
      (filename = Path of a file or variable name,
       name = String or variable name,
       content_type = String or variable name),
      ...
)
```

An example of fileupload_info

```
fileupload_info = (
      (filename = @"c:\tmp\file.txt", name = "file.txt", content_type = "text/plain"),
      (filename = @"c:\tmp\file2.txt", name = "file2.txt", content_type = ct),
      (filename = htmlfile, name = name, content_type = "text/html"),
      (filename = file, name = name, content_type = ct)
)
```

“http_version” and “reason_phrase” are optional.

Effective arguments, which is shown as '〇', and ignored arguments, which is shown as '×', for each method are listed below.

Table 8-72. Methods and arguments of HTTP2

Argument	Methods					
	Get	Put	Post	Delete	Patch	Head
req_header	〇	〇	〇	〇	〇	〇
req_header_file	〇	〇	〇	〇	〇	〇
req_param_multipart	×	〇1	〇1	×	〇1	×
req_param	〇	〇1	〇1	〇	〇1	〇
req_param_file	〇	〇1	〇1	〇	〇1	〇
req_body	×	〇1	〇1	〇	〇1	×
req_body_file	×	〇1	〇1	〇	〇1	×
auth	〇	〇	〇	〇	〇	〇
auth_user	〇	〇	〇	〇	〇	〇
auth_pass	〇	〇	〇	〇	〇	〇
req_cookie	〇	〇	〇	〇	〇	〇
req_cookie_file	〇	〇	〇	〇	〇	〇
req_do_not_base64_attrs	×	〇	〇	×	〇	×
fileupload_info	×	〇	〇	×	〇	×
res_header_filetype	〇	〇	〇	〇	〇	〇
res_header	〇	〇	〇	〇	〇	〇
res_header_file	〇	〇	〇	〇	〇	〇
res_body	〇	〇	〇	〇	〇	〇
res_body_file	〇	〇	〇	〇	〇	〇
res_multipartdata_split	〇	〇	〇	〇	〇	〇
res_cookie_filetype	〇	〇	〇	〇	〇	〇
res_cookie	〇	〇	〇	〇	〇	〇
res_cookie_file	〇	〇	〇	〇	〇	〇

1. For the method 'Put,' 'Post,' or 'Patch,' you can specify either the request body or the parameters, but not both.

The status code is returned to the variable specified in "status_code."

If WinActor.Http2 appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the status code is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

For details and notes, see the subsection “HTTP (advanced)” in the ”User Library Sample Manual.”

8.4.18 Write JSON

```
WinActor.JsonWrite Preamble
 (
  json_table = JSON value,
  write_file   = File path,
  variable     = Variable name to receive the result
  use_file     = true / false // When true is specified, write_file will be adopted.
 )
```

"JSON value" is expressed as a list in which "key<type> = expression" is separated by comma.

"Type" is one of the following. If "Type" is string, object, array or null, specify a value as a string.

If "Type" is Boolean, specify true or false.

Table 8-73. Types for JSON value

Type	Description
integer	Integer
float	Decimal
string	String
object	Object
array	Array
boolean	Boolean
null	Null

The result is stored in the file specified in "write_file" or the variable specified in "variable."

Specify either "write_file" or "variable." If both are specified, "write_file" will be taken when “use_file” is true, .and “variable” when “use_file” is false. If both are specified and “use_file” is omitted, "write_file" will be taken.

If WinActor.JsonWrite appears in an assignment or expression, the result will not be stored in the variable. Specifying the variable to receive the result is optional. When the assign statement or expression appears in the WSS output of a flowchart, the variable to set in the property for the return value is omitted.

Example:

```
i = 128;
ret = WinActor.JsonWrite [name = "Write JSON", comment = ""]
    (
        json_table = ( key_s<string>     = "strvalue",
                       key_i<integer>    = 512,
                       key_v<integer>    = i,
                       key_n<null>       = "",
                       key_b<boolean>    = true,
                       key_a<array>      = "[1, 2, 3]",
                       key_o<object>     = "{ x: 12 }" )
    );
```

The value of variable "ret" after execution:

```
"{
   "key_s" : "strvalue",
   "key_i" : 512,
   "key_v" : 128,
   "key_n" : null,
   "key_b" : true,
   "key_a" : [ 1, 2, 3 ],
   "key_o" : {
      "x" : 12
   }
}"
```

8.4.19 Read JSON

This is to read JSON key values from a file or a variable that stores a JSON object.

```
WinActor.JsonRead Preamble
    (
        json_table = (Key = variable name, ...),
        read_file   = File path string or variable name that stores a file path string,
        variable    = variable name
        use_file    = true / false // When true is specified, read_file will be adopted.
    )
```

Values are read from the file specified in "read_file" or the JSON object stored in the variable specified in "variable."

Specify either "read_file" or "variable." If both are specified, "read_file" will be taken when “use_file” is true, and “variable” when false. If both are specified and “use_file” is omitted, "read_file" will be taken.

For "json_table," specify a key to read a value from the JSON object and a variable name to store the read value.

The types of JSON values are not required.

8.4.20 Other JSON libraries

JSON libraries other than "Write JSON" and "Read JSON" are implemented with the Run Script nodes. See the "Annotation" tab of the property of each Run Script node for details.

8.4.21 Socket

```
WinActor.Socket Preamble
    (
    identifier = variable,                          // connection identifier

    connect = true / false,                         // use connect function or not
    host = String or variable,                      // connection destination
    port = Integer or variable,                     // port number    0 - 65535
    connect_timeout = Integer or variable,          // connection timeout by millisecond
                                                    // integer greater than or equal to -1
    connect_exception_name = String or variable,    // exception name of connection retry

    send = true / false,                            // use send function or not
    send_timeout = Integer or variable,             // send timeout by millisecond,
                                                    // integer greater than or equal to 0
    send_data_type_file = true / false,             // data source is a file or not
    send_data_file = String or variable,            // file path of data source
    send_data_value = String or variable,           // send data
    send_conv_text = true / false,                  // use text conversion or not
    send_input_char = input character encoding,
    send_input_newline = input newline code,
    send_output_char = output character encoding,
    send_output_newline = output newline code,
    send_conv_bin = true / false,                   // convert (base64) to binary data or not
    send_buffer_size = Integer or variable,         // send buffer size
                                                    // integer greater than or equal to 0
    send_data_size = Variable,                      // variable that stores the send data size
```

```
send_shutdown = true / false,                // shutdown sending or not
send_exception_name = String or variable,    // exception name of send retry

receive = true / false,                      // use receive function or not
receive_timeout = Integer or variable,       // receive timeout by millisecond
                                             // integer greater than or equal to 0
receive_data_type_file = true / false,       // received data is stored in file or not
receive_data_file = String or variable,      // file path to store the received data
receive_data_variable = Variable,            // variable to store the received data
receive_conv_text = true / false,            // use text conversion or not
receive_input_char = input character encoding,
receive_input_newline =input newline code,
receive_output_char = output character encoding,
receive_output_newline = output newline code,
receive_conv_bin = true / false,             // convert binary data (to base64) or not
receive_buffer_size = Integer or variable,   // receive buffer size
                                             // integer greater than or equal to 0
receive_end_condition = condition to end reception,
receive_data_size = Integer or variable,     // receive data size
                                             // integer greater than or equal to 0
received_data_size = Variable,               // variable to store the received size
receive_shutdown = true / false,             // shutdown receiving or not
receive_exception_name = String or variable,     // exception name of receive retry

disconnect = true / false,                   // use disconnect function or not
disconnect_timeout = String or variable,     // disconnection timeout by millisecond
                                             // integer greater than or equal to -1
disconnect_exception_name = String or variable       // exception name of disconnection retry
)
```

Set functions to use among the “connect,” “send,” “receive,” and “disconnect.”

Unused functions can also be set, and those settings are stored. When setting a variable for a setting item of an unused function, the variable still needs to be declared. For example, although “send = false, send_data_type_file = true, send_data_value = *variable”* means that the send function is unused and the send data is obtained from a file, the variable still needs to be declared.

When the source of the send data is a variable or a string value, which means "send_data_type_file = false," the only character encoding that can be set to the ‘send_input_char’ is $SOCKET.CharASCII.

When the destination of the received data is a variable, which means "receive_data_type_file = false," the only character encoding that can be set to the ‘receive_output_char’ is $SOCKET.CharASCII.

The connection identifier ‘identifier’ is required to set, but other items are optional. The default values are listed on the table below.

Table 8-74. Default values

Argument type	Default value
true / false	false
String or variable	Empty string
Variable	Anonymous identifier
Port number	0
Timeout	10,000 (ms)
Input/output buffer size	8,192
Input/output character encoding	$SOCKET.CharASCII
Input/output newline code	$SOCKET.NewLineCRLF
Condition to end reception	$SOCKET.EndFin
Receive data size	0
Exception name of retry	$SOCKET.ActionException

However, arguments are required for some items on some conditions.

Table 8-75. Items that require arguments

Item that requires an argument	Condition to require an argument
host	connect = true
port	connect = true
send_data_size	send = true
received_data_size	receive = true
receive_data_size	Receive_end_condition = $SOCKET.EndSize

When “send_data_type_file = true” is set, the file path specified for “send_data_file” by string or via variable indicates the file that is the source of send data. When “send_data_type_file = false” is set, the value specified for “send_data_value” by string or via variable is the send data.

When “receive_data_type_file = true” is set, the file path specified for “receive_data_file” by string or via variable indicates the file that is the destination of received data. When “receive_data_type_file = false” is set, the variable specified for “receive_data_variable” is the destination of received data.

The send text conversion “send_conv_text” and the send binary data conversion “send_conv_bin” cannot be set both true. Also, the receive text conversion “receive_conv_text” and the receive binary data conversion “receive_conv_bin” cannot be set both true.

The character encodings that can be set for “send_input_char,” “send_output_char,” “receive_input_char,” and “receive_output_char” are predefined constants listed on the table below, IANA character set names, and code page numbers. A code page number is specified with an integer from 0 to 65,536. When a character encoding specified by IANA character set or code page number is not supported on the running environment, it is warned and $SOCKET.CharASCII is regarded as the specified encoding.

Table 8-76. Character encodings

Predefined constant	Remarks
$SOCKET.CharASCII	
$SOCKET.CharUTF-16LE	little endian, without BOM
$SOCKET.CharUTF-16LE-BOM	little endian, with BOM
$SOCKET.CharUTF-16BE	big endian, without BOM
$SOCKET.CharUTF-16BE-BOM	big endian, with BOM
$SOCKET.CharUTF-8	without BOM
$SOCKET.CharUTF-8-BOM	with BOM
$SOCKET.CharShift-JIS	Code page 932
$SOCKET.CharEUC-JP	Code page 51932

The newline codes that can be set for “send_input_newline,” “send_output_newline”, “receive_input_newline,” and “receive_output_newline” are predefined constants listed on the table below.

Table 8-77. Newline codes

Predefined constant	Remarks
$SOCKET.NewLineCRLF	
$SOCKET.NewLineCR	
$SOCKET.NewLineLF	
$SOCKET.NewLineNONE	If this is set for “send_output_newline,” or “receive_output_newline,” the newline code at the end of data is removed before sending or storing.

The conditions to end reception that can be set for “receive_end_condition” are predefined constants listed on the table below.

Table 8-78. Conditions to end reception

Predefined constant	Remarks
$SOCKET.EndFin	until FIN is received
$SOCKET.EndSize	until the number of octets specified for “receive_data_size” is attained
$SOCKET.EndEmpty	until no readable data remains

The retry exception names that can be set for ‘connect_exception_name,’ ‘send_exception_name,’ ‘receive_exception_name,’ and ‘disconnect_exception_name’ are $SOCKET.ActionException and other strings of exception names. When $SOCKET.ActionException is specified, “アクション例外” (Japanese) or “ActionException” (English) is used depending on the language setting at the time of loading the WSS scenario.

If an empty string is set as an exception name, no exception is raised and the scenario continues.

8.5 Libraries not listed in the adapter actions

Libraries not described in the adapter actions are implemented with the Run Script nodes or subroutines.

These libraries can be used by placing them in WSS-enabled scenarios in WinActor.

By saving the created WSS-enabled scenario in a file, these libraries will be saved in wss7 and uss7 files.

9. Notes on restoration of expressions

In the WinActor scenarios loaded from the WSS, translated expressions may include some additional working variables and nodes. Saving such a scenario back to the WSS, original expressions were divided and became ugly to read through

To mitigate this problem, mechanisms to remove working variables and additional nodes and to restore expressions to the near original state have been introduced.

However, if a working variable is edited and reused by a user on the “Flowchart” area of WinActor, the expressions restored on the WSS may look far different from the expressions the user intends. Therefore, the working variables that meet any of following conditions are left intact on the restored WSS.

- Conditions to leave the working variable or the additional node intact
  - A node which refers to the working variable is added.
  - The working variable is newly referred to read a value.
    - Referrer: All inputs where “Value or variable”, or “Variable” is required.
  - A node to assign something to the working variable is added.
  - The working variable is newly set to the destination of assigning a value
    - Destination: the destination variable of the following return of a node.
      - the returned variable $...$ from VB in the “Run Script” node.
      - the return value of the “Call Subroutine” node.
      - the return value of the “Exit Subroutine” node.
      - the return value of the “Call Scenario File” node.
      - the return value of the “Exit Scenario” node.
  - A sticky note is attached to the node which is added at the time of loading the WSS.
  - A breakpoint is set at the node which is added at the time of loading the WSS.
  - The working variable set as a local variable in the “Call Subroutine” node is assigned an initial value or a variable in the property of the node.
  - The working variable is deleted from the “Local variable list (variables to restore the initial value at the end)” settings in the property of the “Subroutine Group” node.
  - The working variable is specified in the return setting of the “Call Scenario File” node.

- Conditions to leave working variables intact without user modifications
  - The working variable occurred in the “conditional expression” of the “Loop condition” of the “while” or the “dowhile” statement.
  - The working variable occurred in the “End” expression of the “Loop condition” of the “while” or the “dowhile” statement.
  - If the “conditional expression” of the “Case statement” in the “Switch statement” has working variables and has been converted to the “IF statement” (the “Decision” node on WinActor), the “Switch statement” cannot be restored.
  - The working variable occurred in an anonymous subroutine.
  - If a working variable used in a called scenario is specified in the “Variables inherited from the destination” of the “Call Scenario File” node, it may not be inherited when the working variable is deleted in the restoration process.
  - The working variables which was in the “Floating part” of the original WSS remains if the part is converted to a subroutine on the scenario of WinActor.

WinActor®

Programming Language
WinActor Scenario Script

NTT ADVANCED TECHNOLOGY CORPORATION

WA7-U-20250905