プログラミング言語
WinActor Scenario Script

NTTアドバンステクノロジ株式会社

Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION

商標について

本書において以下に記載された名称、およびその他記載されている会社名、製品名は、各社の登録商標または商標です。なお、本文中では ™、®、©マークは省略しています。

- WinActor は NTT アドバンステクノロジ株式会社の登録商標です。

- Microsoft、Windows※1、Internet Explorer、Excel、VBScript※2 は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

  ※1　Windows の正式名称は、Microsoft Windows Operating System です。

  ※2　VBScript の正式名称は、Microsoft Visual Basic Scripting Edition です。

- その他の記載されている会社名、製品名は各社の商標または登録商標です。

本書について

この「プログラミング言語 WinActor Scenario Script」（以下、本書）は、WinActor のシナリオを記述できるプログラミング言語 WinActor Scenario Script の説明書です。

本書は、WinActor のシナリオを GUI ではなく、プログラムで記述可能な上級プログラマを対象としています。

本書に関する注意

- 本書および提供するソフトウェア類に付された著作権表示「Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION」の変更、削除をすることはできません。
  本書の著作権は NTT 株式会社及び NTT アドバンステクノロジ株式会社に帰属します。
- 本書では、Windows の操作方法や機能を理解されていることを前提として説明しています。本書に記載されていないことについては、Microsoft が提供しているドキュメントなどをご覧ください。
- 本書では、Windows 10 の使用を前提として説明しています。

目次

商標について i
本書について ii
目次 iii

1. はじめに 1

2. 用語 2

3. 記号と記号列 3

4. データ型 5

4.1 コメント 5
4.2 数値 5
4.3 論理値 5
4.4 識別子 6
4.5 文字列 19
4.6 構造体 20
4.6.1 プリアンブル 20
4.6.2 アダプタ引数リスト 21
4.6.3 無評価タプル 22
4.6.4 定数タプル 22

5. シナリオ 24

5.1 シナリオの構成 24
5.2 変数部 24
5.2.1 変数宣言 24
5.3 ウィンドウマッチルール部 25
5.3.1 window_title の定数タプル 26
5.3.2 window_class の定数タプル 26
5.3.3 process_name の定数タプル 26
5.3.4 window_size の定数タプル 27
5.3.5 ウィンドウマッチルール部の例 27
5.4 メイン部 28
5.5 フローテイング部 28
5.6 サブルーチン部 30
5.7 監視ルール部 30
5.8 イベント監視部 31
5.9 ブレークポイント情報部 34
5.10 シナリオ情報部 35

5.11 イメージ部 36
5.11.1 イメージ宣言 37
5.11.2 イメージ宣言の例 37
5.12 フローチャート情報部 37
5.13 単語辞書部 38
6. 文（statement） 39
6.1 説明 39
6.2 グループ文（group statement） 39
6.3 if 文 39
6.4 while 文 40
6.4.1 ループ条件 40
6.5 dowhile 文 41
6.5.1 ループ条件 42
6.6 switch 文 42
6.6.1 ケース文 42
6.6.2 デフォルト文 42
6.7 try 文 43
6.7.1 キャッチ文 43
6.8 return 文 43
6.9 シナリオ終了文 44
6.10 break 文 44
6.11 continue 文 44
6.12 サブルーチン呼び出し文 44
6.13 シナリオファイル呼び出し文 45
6.14 アダプタアクション文 46
6.15 代入文 46
6.16 四則演算 47
7. 式（expression） 48
7.1 因子（factor） 48
7.2 定数式 49
7.2.1 定数式用二項演算子 49
7.2.2 定数因子 50
7.3 条件式 50
7.3.1 条件式用二項演算子 51
7.3.2 条件式因子 51
8. アダプタアクション 52
8.1 自動記録アクション 52
8.1.1 イベント記録 クリック 52
8.1.2 イベント記録 文字列設定 53

8.1.3 イベント記録 リスト選択 54
8.1.4 イベント記録 タブ選択 55
8.1.5 イベント記録 エミュレーション 55
8.1.6 イベント記録 文字列取得 58
8.1.7 イベント記録 リスト取得 59
8.1.8 イベント記録 チェック状態取得 59
8.1.9 イベント記録 有効無効状態取得 60
8.1.10 イベント記録 リスト一括取得 60
8.1.11 UIAutomation 61
8.1.12 UIAutomation ライブラリ 66
8.1.13 UIAutomation ダンプ 67
8.2 自動記録アクション（IE） 68
8.2.1 IE モード記録 クリック 68
8.2.2 IE モード記録 文字列設定 70
8.2.3 IE モード記録 リスト選択 70
8.2.4 IE モード記録 文字列取得 71
8.2.5 IE モード記録 リスト取得 71
8.2.6 IE モード記録 チェック状態取得 72
8.2.7 IE モード記録 有効無効状態取得 72
8.2.8 IE モード記録 表の値取得 73
8.2.9 IE モード記録 リスト一括取得 74
8.3 アクション、ユーザ、変数 74
8.3.1 画像マッチング 74
8.3.2 輪郭マッチング 77
8.3.3 OCR マッチング 80
8.3.4 ウィンドウ状態待機 82
8.3.5 指定時間待機 83
8.3.6 文字列送信 84
8.3.7 コマンド実行 85
8.3.8 スクリプト実行 85
8.3.9 Python 実行 87
8.3.10 Excel 操作 90
8.3.11 クリップボード 90
8.3.12 クリップボードに設定 91
8.3.13 クリップボードの値を取得 91
8.3.14 待機ボックス 92
8.3.15 インプットボックス 92
8.3.16 選択ボックス 92
8.3.17 音（ブザー） 93
8.3.18 音（WAVE ファイル） 93
8.3.19 変数値設定 SetVariable 93
8.3.20 変数値コピー CopyVariable 94
8.3.21 日時取得 95
8.3.22 ユーザ名取得 96
8.3.23 四則演算 96
8.3.24 カウントアップ 97

8.3.25 全角化/半角化 97
8.3.26 イベント監視 98
8.3.27 イベント監視登録 98
8.3.28 イベント監視解除 98
8.3.29 イベント監視終了 98
8.4 WinActor メール管理、HTTP 関連、JSON 99
8.4.1 メール受信設定 99
8.4.2 メール受信 100
8.4.3 メール選択 101
8.4.4 選択中メールの状態変更 102
8.4.5 メールフォルダ同期 102
8.4.6 メール処理済み削除 103
8.4.7 メール削除 103
8.4.8 メール情報コピー 103
8.4.9 添付ファイル名取得 104
8.4.10 メール情報取得 104
8.4.11 メール受信設定インポート 104
8.4.12 Gmail 受信設定 105
8.4.13 Gmail 受信 106
8.4.14 Gmail 送信設定 107
8.4.15 Gmail 送信 107
8.4.16 HTTP 108
8.4.17 HTTP（詳細） 109
8.4.18 JSON 形式書き込み 113
8.4.19 JSON 形式読み取り 115
8.4.20 その他の JSON ライブラリ 115
8.4.21 Socket 通信 115
8.5 アダプタアクションに記載のないライブラリ 120
9. 式復元機能の注意事項 121

1.　はじめに

プログラミング言語 WinActor Scenario Script（以下、WSS）は、WinActor Ver.7 のシナリオを、上級プログラマ向けに手続き型言語に変換したものである。

WSS で記述したファイルは、テキストエディタで編集可能で、拡張子.wss7 を付けて保存し、wss7 ファイルと呼ぶ。

wss7 ファイルは、同時に生成される uss7 ファイルと併せて WinActor に読み込むことで、シナリオに戻すことができる。

WinActor でシナリオをファイルに保存する手順、またはファイルを読み込んでシナリオに戻す手順については、『WinActor 操作マニュアル』を参照のこと。

WSS は、C 言語に似た手続き型言語であるが、WinActor のシナリオと密接に対応しているため、いくぶん自由度が少ない。

2. 用語

以下の記述で使用する用語をあげる。詳細は後述する。

表 2-1 用語

用語	備考
フローチャート	WinActor のフローチャート表示エリアに表示されるフローチャートを指すのに使用する。
識別子（identifier）	
値を持つていない名前	
予約語（reserved identifier）	
アクション名（action name）	
サブルーチン名（subroutine name）	
JSON 型名（JSON type name）	JsonWrite アダプタに出現
値と関連づく名前	
読み書き可能なものと読み込み専用のものがある	
単純識別子（simple identifier）	
数名識別子（number identifier）	
特殊識別子（special identifier）	
コンパイラ生成作業用識別子（compiler generated working identifier）	
読み込み専用のものだけ	
既定義定数名（predefined constant identifier）	
代入の形をしているが、変数ではないもの	プリアンブル、定数タプルに出現
属性名（attribute identifier）	
文字列表記（string literal）	
単純文字列（simple string literal）	
逐語的文字列（verbatim string literal）	
ブロック逐語的文字列（block verbatim string literal）	
構造体（structure）	
プリアンブル（preamble）	
アダプタ引数リスト（adapter parameter list）	
無評価タプル（verbatim tuple）	
定数タプル（const tuple）	

3. 記号と記号列

WSS で意味を持つ記号と記号列をあげる。

表 3-1 記号と記号列

記号と記号列	意味	使用箇所など
(	左丸括弧	
)	右丸括弧	
⌊	左角括弧	
⌋	右角弧括	
{	左波括弧	
}	右波括弧	
@(	アット左丸括弧	無評価タプルに使用
=	等号	
+	加算符号	四則演算子 単項演算子 定数式用二項演算子
−	減算符号	四則演算子 単項演算子 定数式用二項演算子
*	乗算符号	四則演算子 定数式用二項演算子
/	除算符号	四則演算子 定数式用二項演算子
+^	加算符号（整数演算）	四則演算子 単項演算子 定数式用二項演算子
−^	減算符号（整数演算）	四則演算子 単項演算子 定数式用二項演算子
*^	乗算符号（整数演算）	四則演算子 定数式用二項演算子
/^	除算符号（整数演算）	四則演算子 定数式用二項演算子
==	等しい	定数式用二項演算子 条件式用二項演算子
!=	等しくない	定数式用二項演算子 条件式用二項演算子
>=	より大きいまたは等しい	定数式用二項演算子 条件式用二項演算子
>	より大きい	定数式用二項演算子 条件式用二項演算子
<	より小さい	定数式用二項演算子 条件式用二項演算子
<=	より小さいまたは等しい	定数式用二項演算子 条件式用二項演算子
!	否定	定数式用単項演算子 条件式用単項演算子
&&	かつ	定数式用二項演算子
||	または	定数式用二項演算子
~	正規表現マッチ	定数式用二項演算子 条件式用二項演算子
!~	正規表現アンマッチ	定数式用二項演算子 条件式用二項演算子

記号と記号列	意味	使用箇所など
’	クオート	識別子用
_	下線	識別子用
″	ダブルクオート	単純文字列用
@″	逐語的文字列開始記号	
@″″″	ブロック逐語的文字列開始記号	
@″″″″	ブロック逐語的文字列開始記号	
.	ピリオド	
,	カンマ	
;	セミコロン	
$	ドル記号	数名識別子 特殊識別子 既定義定数に使用
/*	コメント開始	
*/	コメント終了	
//	行コメント開始	
\	エスケープ記号	文字列の記法を参照のこと
0 1 2 3 4 5 6 7 8 9	数字	

4. データ型

4.1 コメント

/* から */ で囲まれた部分はコメントとして読み飛ばす。複数行にわたっても構わない

// から行末までもコメントとして読み飛ばす。

4.2 数値

数値には整数と浮動小数があり、10 進法で表わす。

表 4-1 数値の構成

数値	構成
整数	数字列
浮動小数	数字列.数字列
	数字列 e+-数字列
	数字列.数字列 e+-数字列

e は 10 のべきを意味する。

e は大文字の E でも構わない。

e の直後には+または-を書くことができる。

e の直後の+または-は省略できる。

e の右の数字列は省略できない。

数値の表記の両側を半角のダブルクオートで囲んだ文字列も、数値として式や定数式に使用できる。

全角の数字＋－Ｅ ｅピリオドも数値の表記に使用できる。

カンマ文字による桁区切りを含む数字列を半角のダブルクオートで囲んだ単純文字列は、カンマ桁区切りの数値表記と呼び、数値として式や定数式に使用できる。

カンマ桁区切りの数値表記には、全角の数字や全角のカンマ「,」も使用できる。

カンマ桁区切りの数値表記は、ピリオドや e を含んでいなくても常に浮動小数となる。整数演算では実行時エラーとなるので注意すること。

4.3 論理値

TRUE または FALSE。文字列の "TRUE", "FALSE" も論理値として働く。

4.4　識別子

識別子は大域的である。サブルーチン内部の変数も変数部で宣言する必要がある。

予約語と既定義定数は大文字小文字を区別しない。

それら以外の識別子は大文字小文字を区別する。

■　識別子の記法

表 4-2 識別子の記法

識別子の記法	説明
[_英数字]+	英数字と下線だけの識別子。(単純識別子)
’.*’	シングルクオートで任意の文字を囲む。 日本語の入った名前を作成する際に使用する。 特に '' （連続した 2 つのシングルクオート）は、無名識別子（anonymous）として、値を省略して既定値を使うことを指示するような場合に活用できる。
$[_-英数字ピリオド]+	先頭が $ で、英数字および下線、ハイフン、ピリオドからなる識別子。 特殊識別子や既定義定数名に使用される。
$[数字]+	上に含まれるが、$数字列を数名識別子と呼ぶ。 フローチャート上では $ を取り除いた名前として扱われる。
__work_[0-9][0-9][0-9][0-9]	__work_ に 4 桁の数字が付いた名前。コンパイラが作業用に生成した識別子。 変更された場合の動作は保証しない。 変数グループ __internal__ に生成する。

いずれの記法もバックスラッシュ文字は次の文字をエスケープする

■　予約語

以下の識別子は予約語であって、プログラム構文に使用する。
大文字と小文字は区別しない。
構文で使用できる位置は決まっている。
それ以外の箇所では、単純識別子として変数名に使っても構わないが、サブルーチン名として予約語を指定したときの動作は保証しない。
各予約語の詳細は後述する。

表 4-3 予約語

名前
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
if
then
else

名前
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

■　特殊識別子

WinActor の特殊変数は、式に使用できる。
先頭の ＄ を含めてすべて大文字で記述すること。
名前に - が含まれているものがあるが、WinActor の特殊変数名に一致する名前であれば、- も含めて名前とし、減算記号とは解釈しない。
使用できる特殊変数は、『WinActor 操作マニュアル』の『特殊変数』を参照のこと。
読み込み専用となっている特殊変数に代入はできない。

■　既定義定数名

以下の識別子を既定義定数名という。
大文字小文字を区別しない。
アクションのモード指定など、数や文字列で指定するものを名前で記述するために用意してある。
読み込み専用。

式、および、定数式に使用できる。

表 4-4 既定義定数名

名前	値
$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
$IE.WaitSettingOption	"specified_option_info"
$IE.WaitSettingScenario	"specified_scenario_info"
$IE.WaitSettingNode	"specified_node"
$IEGetTableInfo.GetCell	"getcell"
$IEGetTableInfo.ExistCell	"existcell"

名前	値
$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
$ImageMatch.LeftClickDrag	9
$ImageMatch.RightClickDrag	10
$ImageMatch.Same	0
$ImageMatch.Half	1
$ImageMatch.Quarter	2

名前	値
$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"
$OutlineMatch.Path_File	"FilePath"
$OutlineMatch.Path_Folder	"FolderPath"
$OCRMatch.Check	1
$OCRMatch.LeftClick	2
$OCRMatch.RightClick	3

名前	値
$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"
$MailCopyClip.UniqueID	"UID"
$MailCopyClip.FolderName	"DIR"
$MailCopyClip.Status	"STAT"
$MailCopyClip.SendDate	"SEND_DATE"
$MailCopyClip.From	"FROM"

名前	値
$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"
$UIA.WaitSettingOption	"option"
$UIA.WaitSettingScenario	"scenario"
$UIA.WaitSettingNode	"node"
$UIA.WaitForWindow	"window"
$UIA.WaitForControl	"element"

名前	値
$UIA.CommonPattern	"CommonPattern"
$UIA.ExpandCollapsePattern	"ExpandCollapsePattern"
$UIA.InvokePattern	"InvokePattern"
$UIA.ScrollPattern	"ScrollPattern"
$UIA.SelectionPattern	"SelectionPattern"
$UIA.SelectionItemPattern	"SelectionItemPattern"
$UIA.TogglePattern	"TogglePattern"
$UIA.ValuePattern	"ValuePattern"
$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"
$UIA.SelectOne	"SelectOne"
$UIA.Toggle	"Toggle"
$UIA.GetToggleState	"GetToggleState"
$UIA.IsReadOnly	"IsReadOnly"
$UIA.GetValue	"GetValue"
$UIA.SetValue	"SetValue"

名前	値
$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"
$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"

名前	値
$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"

■　アクション名

アクション名を表す識別子は以下のとおり。大文字小文字を区別しない。

表 4-5 アクション名

名前
ClickWin32
SetTextWin32
SelectListWin32
SelectTabWin32
EmulationWin32
GetTextWin32
GetListWin32
GetCheckWin32
GetEnableWin32

名前
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
Beep
Speaker
Launcher
Clipboard
SetToClipboard
GetFromClipboard
Script
TextConvert
Excel
MailReceive
MailSelect

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

4.5　文字列

■　文字列の表記

表 4-6 文字列の表記

名前
単純文字列（simple string literal）
逐語的文字列（verbatim string literal）
ブロック逐語的文字列（block verbatim string literal）

■　文字列の記法

表 4-7 文字列の記法

記法	説明
″.*″	ダブルクオートで文字を囲んだもの。単純文字列。 複数行にわたっても構わない。 \（バックスラッシュ文字）はエスケープ記号であり、その次の文字と併せて次の意味をもつ。 複数行にわたる文字列の場合、行末のエスケープ記号は無視する。行末にエスケープ記号を書いても次の行とは連結されない。 \\　バックスラッシュ文字 \0　nul \b　backspace \r　return \n　linefeed \f　formfeed \t　horizontal tab \v　vertical tab
@″.*″	@" ではじめて " で終わる。逐語的文字列。 バックスラッシュ文字は、特別な解釈はされず、そのままバックスラッシュ文字として扱われる。 文字列内にダブルクオートを含めるには "" （連続した 2 つのダブルクオート）と記述する。 ファイルやフォルダのパス名に使用することを想定している。
@″″″[\s]*$ ... ″″″	行が @""" で終わったらその次の行から、""" だけが現れる行までの文字をすべて文字列として扱う。ブロック逐語的文字列。 一切のエスケープ処理をしない。 スクリプトアダプタのノートやスクリプトの記述に使用することを想定している。

記法	説明
@″″″″[\s]*$ ... ^″″″″[\s]*, ?[\s]*$	行が @""""""" で終わったらその次の行から、""""""" だけ（1 つのカンマは許容）が現れる行までの文字をすべて文字列として扱う。ブロック逐語的文字列。 一切のエスケープ処理をしない。 スクリプトアダプタのノートや python スクリプトの記述に使用することを想定している。ノートや python スクリプトのコメントにこの記法の開始または終わりの文字列が含まれているときの動作は保証しない。

4.6　構造体

4.6.1　プリアンブル

ノードのプロパティの「名前」や「コメント」を記述する際に使用する。

isclosed 属性を true に設定すると、フローチャートではノードは閉じて表示される。

特に、ID 属性は、付箋の張り付け先ノード、ブレークポイント設定ノードの対応をとるために使用する。

ID で対応をとるノードについては後述する。

プリアンブルを置ける位置は決まっている。

プリアンブルは省略可能。

省略した場合、フローチャートでのノードの名前はノードごとの固定の名前になり、コメントは空になる。

定数式の詳細は後述する。

■　プリアンブルの記法

```
[ 属性名 = 定数式, ... ]
```

左角かっこと右角かっこ内に、属性名 = 定数式 をカンマで区切って並べる。

属性名 = 定数式 は 0 個でもよい。

意味のある属性名はプリアンブルを置く場所ごとに定まっている。

ノードに付与するプリアンブルの属性には、通常は name, comment 属性がある。

name はプロパティの名前、comment はコメント欄に対応している。

個々のプリアンブルに特有の属性については後述する。

■　プリアンブルの例

```
[ name = ″分岐グループ″, comment = ″入力ごとの分岐″, isclosed = true ]
```

4.6.2　アダプタ引数リスト

■　アダプタ引数リストの記法

```
( 属性名|文字列<識別子> = 実引数 , ... )
```

(ではじまり、) で終わる。ここにカンマで区切って並べる。( ) だけのものもある。

「<識別子>」は特定のアダプタでのみ意味を持つ。(後述)

「属性名|文字列<名前> = 」部分は使用箇所によっては省略可能。

「実引数」部分には次の要素を記述する。条件式は書けない。

表 4-8 実引数の要素

実引数の要素	備考
式	アダプタの属性によっては式は許されない場合がある。(後述)
定数式	
無評価タプル	
アダプタ引数リスト	

■　アダプタ引数リストの例

例 1: WinActor.SendText アダプタの例

```
(
    window_rule_ref = "無題-メモ帳",
    control = (instance<true> = 0, text<true> = "無題 - メモ帳", position<true> = ret01),
    value = var01,
    sendcr = true,
    verify = true,
    capture = (imageid = "img_20190613172532792", x = 409, y = 10)
      // 実引数にアダプタ引数リストを使用した例
)
```

例 2: WinActor.EmulationWin32 アダプタの引数リストの例

```
(
    window_rule_ref = "Window",
    action = @(Wait, 300),      // 無評価タプルの例
    capture = (imageid = "_", x = 0, y = 0)
)
```

例 3: WinActor.SelectBox アダプタの引数リストの例

```
(
```

```
    message = "選んで",
    items = ("赤", "青", "白")  // 「属性名 =」を省略する例
)
```

4.6.3 無評価タプル

識別子、文字列、数値 を並べた構造体。

文字列や数値を得るために定数式を含んでもよい。

エミュレーションアダプタのマウスアクションの記述を想定している。

■ 無評価タプルの記法

```
@( 識別子|文字列|数値|(定数式), ... )
```

@(ではじまり、) で終わる。ここにカンマで区切って、識別子または文字列または数値を記述する。

定数式は (と) で囲む。囲まないとシンタクスエラーになる。

識別子は単に名前として扱い、定数宣言されていてもその値は使わない。

定数宣言については後述する。

■ 無評価タプルの例

```
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, ('短い待ち')))
```

「短い待ち」は定数宣言で数値が設定されているとする

4.6.4 定数タプル

以下の場所で使用する。名前に定数を与える。

表 4-9 定数タプルの使用場所

定数タプルの使用場所
ウィンドウマッチルール部

定数タプルの使用場所
フローテイング部の付箋
ブレークポイント情報部
シナリオ情報部
イメージ部
フローチャート情報部

■　定数タプルの記法

```
( 属性名|文字列 = 定数式 , ... )
```

(ではじまり、）で終わる。ここにカンマで区切って定数式を並べる。

左辺の名前は属性名、または、文字列で記述するが、文字列が同じなら同じ名前とみなす。
(例 abc と "abc" は同じ)

属性名、文字列は重複した場合、エラーとなる。

5. シナリオ

5.1 シナリオの構成

シナリオは複数種類の部で構成され、下記の順に記述する必要がある。
部によって必要な個数が決められている。

表 5-1 シナリオの構成

No.	名前	必要な個数
❶	変数部	0 個から複数個
❷	ウィンドウマッチルール部	0 個から複数個
❸	メイン部	1 個
❹	フローテイング部	0 個から複数個
❺	サブルーチン部	0 個から複数個
❻	監視ルール部	0 個 または 1 個
❼	ブレークポイント情報部	0 個 または 1 個
❽	シナリオ情報部	1 個
❾	イメージ部	0 個 または 1 個
❿	フローチャート情報部	1 個
⓫	単語辞書部	0 個 または 1 個

5.2 変数部

■ 構文

```
Var_group グループ名 = ( 変数宣言 , ... )
```

■ 説明

変数宣言と定数宣言を記述する。
変数をグループにまとめるために複数記述可能。

グループ名は文字列、または識別子で記述する。
グループ名 = は省略可能。
変数宣言はカンマで区切って 0 個以上記述可能。

5.2.1 変数宣言

■ 構文

```
const 識別子 = 定数式 プリアンブル
```

■ 説明

変数宣言と定数宣言が可能。

const を省略すると変数宣言、const を付与すると定数宣言となる。

変数宣言の場合、= 定数式 は省略可能。記述したときには初期値となる。

定数宣言の場合、初期値は必須。

const を付与した識別子に代入はできない。

const を付与した識別子は定数式内で使用できる。

識別子はシナリオ内で一意であること。別の変数グループでも同じになってはいけない。

WinActor の変数一覧でマスク欄にチェックがある変数の初期値は、wss7 ファイルには出力されない。

また、プリアンブルに mask = true が出力される。

変数宣言のプリアンブルに mask = true があり、初期値が省略されている場合、WinActor は uss7 ファイルに保持してある初期値を使用する。

5.3 ウィンドウマッチルール部

■ 構文

```
Window_rule ウィンドウ識別名 プリアンブル = (
    window_title = 定数タプル,
    window_class = 定数タプル,
    process_name = 定数タプル,
    window_size  = 定数タプル
)
```

■ 説明

ウィンドウマッチルール部は、WinActor 本体で生成させて、WSS では必要に応じてルールを編集するといった使い方を想定している。
ウィンドウ識別名 は文字列。ウィンドウマッチルール部内で重複した場合はエラーとなる。
プリアンブルで有効な属性は、comment のみである。

5.3.1　window_title の定数タプル

■　構文

```
(original_value = キャプチャ時タイトル名文字列, pattern = 照合用タイトル名文字列, rule = ルール)
```

■　ルールに指定できる既定義定数

表 5-2 window_title のルール

既定義定数	説明
$WindowRule.Unspecified	指定せず
$WindowRule.ExactMatch	完全一致
$WindowRule.PartialMatch	部分一致
$WindowRule.AtFirst	前部分一致
$WindowRule.AtLast	後部分一致
$WindowRule.Regex	正規表現指定

5.3.2　window_class の定数タプル

■　構文

```
(original_value = キャプチャ時クラス名文字列, pattern = 照合用クラス名文字列, rule = ルール)
```

■　ルールに指定できる既定義定数

表 5-3 window_class のルール

既定義定数	説明
$WindowRule.Unspecified	指定せず
$WindowRule.ExactMatch	完全一致

5.3.3　process_name の定数タプル

■　構文

```
(original_value = キャプチャ時プロセス名文字列, pattern = 照合用プロセス名文字列, rule = ルール)
```

■　ルールに指定できる既定義定数

表 5-4 process_name のルール

既定義定数	説明
$WindowRule.Unspecified	指定せず
$WindowRule.ExactMatch	完全一致

5.3.4　window_size の定数タプル

■　構文

```
(original_value = キャプチャ時ウィンドウサイズ文字列, pattern = 照合用ウィンドウサイズ文字列, rule = ルール)
```

キャプチャ時ウィンドウサイズ文字列 と 照合用ウィンドウサイズ文字列 は、幅のサイズおよび高さのサイズをカンマで連結した文字列である。カンマ桁区切りの数値表記の形をなしているが、ここでは 2 つの数値であり、1 つの数値として演算することはできない。

■　ルールに指定できる既定義定数

表 5-5 window_size のルール

既定義定数	説明
$WindowRule.Unspecified	指定せず
$WindowRule.Equal	一致
$WindowRule.GTE	大きいか等しい
$WindowRule.LTE	小さいか等しい

5.3.5　ウィンドウマッチルール部の例

```
Window_rule "無題-メモ帳" [comment = ""] = (
    window_title = (original_value = "無題 - メモ帳", pattern = "無題 - メモ帳", 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　メイン部

■　構文

```
main プリアンブル {
     文の並び
}
```

■　説明

フローチャートのシナリオの開始から終了に該当する部。
メイン部に記述された文が、並びの順序で実行される。
メイン部のプリアンブルにおいて有効な属性はない。

5.5　フローテイング部

■　構文

```
floating プリアンブル
{
   tag プリアンブル
   ( tag_comment = コメント文字列, target = 定数式, area = 相対位置 );
}

または

floating プリアンブル
{
   文の並び
}
```

■　説明

フローテイングには 1 つの付箋（tag 設定）、または、複数の文を記述することができる。

付箋をノードに紐付けるにはノードのプリアンブルに ID 属性を記載し、tag の target 属性でノードの ID 番号を指定する。
ID 番号は整数、または、空文字列、または、整数となる文字列でなければならない。
空文字列を指定したときは、ノードとの関連付けのない独立した付箋の指定となる。
tag_comment 属性、target 属性はどちらも省略可能で、空文字列を指定したとみなす。

area 属性は付箋の相対位置を指定する。相対位置には下表のいずれかを指定する。
area 属性は省略可能で、省略した場合は付箋の絶対位置指定となる。
なお、area 属性はフローチャート上では付箋の位置で判定するので、プルダウンメニューなどはない。

表 5-6 付箋の相対位置

相対位置	説明
$TAG.TopLeft	上部左側
$TAG.TopCenter	上部中央
$TAG.TopRight	上部右側
$TAG.CenterLeft	上下中央左側
$TAG.CenterCenter	上下左右中央
$TAG.CenterRight	上下中央右側
$TAG.BottomLeft	下部左側
$TAG.BottomCenter	下部中央
$TAG.BottomRight	下部右側

文の並びが複数となるときには、コンパイラがグループノードを生成して、複数の文をまとめる。

floating のプリアンブルに、フローチャートでのタブ（tab_id_ref）とノード位置（x, y）を記述する。

tab_id_ref にはフローチャート情報部の tab_id に指定した文字列を与える。

tag のプリアンブルには名前（name）とコメント（comment）が記述できる。

文を記述した floating のプリアンブルにはタブとノード位置に加えて ID と付箋の非表示指定（TagVisible）ができる。

floating ノードの付箋を表示したくないときには、 TagVisible = false を指定する。

■ フローテイング部の例

```
floating [x = 216, y = 37.5, tab_id_ref = 0]
{
    tag [name = "付箋", comment = "TimerWait の付箋"]
    (tag_comment = "独立の待機", target = 80);          // target = "80" も可。
}

floating [ID = 80, x = 1305, y = 6, tab_id_ref = 0]
{
    WinActor.TimerWait [name = "独立配置の待機", comment = ""]
    (   mode = $TimerWait.Sleep,
        timeout = 10,
        date_format = $TimerWait.ScenarioInfoDateFormat,
        timezone = $TimerWait.ScenarioInfoTimeZone
```

```
    );
}
```

5.6　サブルーチン部

■　構文

```
sub サブルーチン名 プリアンブル
    localvars( 変数名の並び ),
    chkempty( true または false )
{
   文の並び
}
```

■　説明

サブルーチン名 は文字列、または、識別子で記述する。名前の重複はエラーとなる。
プリアンブルには、フローチャートでのタブ（ tab_id_ref 属性）とノード位置（ x 属性, y 属性）、および、折りたたみ状態（ isclosed 属性、値は true または false ）を記述する。name 属性は、無効である。ノードの付箋を表示したくないときには、 TagVisible = false を指定する。

chkempty(true または false) は省略可能。省略する場合には、その直前のカンマも書かないこと。
localvars の変数名は、変数部で変数として宣言されていなければならない。
サブルーチンの実行が終わったあとに、実行前の値に戻される。

一部のライブラリではサブルーチンの 文の並び が秘匿されている。秘匿された 文の並びは、WSS では表示されない。変更することもできない。

5.7　監視ルール部

■　構文

```
Rules = (
    (window_rule_ref = Window_rule ウィンドウ識別名, 監視アクション), ...
)
```

■　監視アクション

次の 3 つのいずれか。

表 5-7 監視アクション

監視アクション	説明
throw(例外名)	例外を発生させる。例外名は文字列

監視アクション	説明
subref(サブルーチン名)	サブルーチンを実行させる。 サブルーチン名は、文字列または識別子
error	シナリオを停止させる

■　説明

ウィンドウ識別名は、ウィンドウマッチルール部（Window_rule）で宣言されていなければならない。
ウィンドウマッチルールに当てはまるウィンドウが表示されたときに監視アクションが実行される。
Rules には複数のウィンドウ識別名と監視アクションの組を記述することができる。

■　監視ルール部の例

```
Rules = (
    (window_rule_ref = "警告", throw("警告発生")),
    (window_rule_ref = "ネットワーク資格情報入力", subref("ブザー音を鳴らす")),
    (window_rule_ref = "侵入禁止", error)
)
```

5.8　イベント監視部

■　構文

```
Events = (
  イベント監視名 プリアンブル = (
    trigger = イベントトリガー条件,
    イベントトリガー条件毎パラメータ,
    呼び出し処理毎パラメータ,
    from_the_start = true または false    // 初期監視有無。監視対象とするかの初期値
  ), ...
```

■　イベントトリガー条件

イベントトリガー条件には以下のいずれかを指定。

表 5-8 イベントトリガー条件

条件	説明
$EVENT.UpdateFile	特定ファイルの更新
$EVENT.UpdateFolder	特定フォルダの更新
$EVENT.SpecifiedTime	時間（指定時間）
$EVENT.Monthly	時間（毎月）
$EVENT.Weekly	時間（毎週）

条件	説明
$EVENT.Everyday	時間（毎日）
$EVENT.Hour	時間（毎時）
$EVENT.Minute	時間（毎分）
$EVENT.WindowState	ウィンドウ状態
$EVENT.Mail	メール受信

■　イベントトリガー条件毎パラメータ

イベントトリガ条件によって設定値が異なる箇所について記載。

特定ファイルの更新の場合

```
path = イベント監視対象のファイルパス,
```

特定フォルダの更新の場合

```
path = イベント監視対象のフォルダパス,
```

時間（指定時間）の場合

```
datetime = 年月日時分秒      // yyyy/MM/dd HH:mm:ss 形式
```

時間（毎月）の場合

```
type = 月指定方法の識別子,
day = 日,          // dd 形式。月指定方法の識別子が毎月（指定日）の場合のみ有効
hour = 時間,       // HH 形式
minute = 分,       // mm 形式
```

月指定方法の識別子には以下のいずれかを指定。

表 5-9 月指定方法の識別子

識別子	説明
$EVENT.DAY	毎月（指定日）
$EVENT.STARTDAY	毎月初
$EVENT.LASTDAY	毎月末

時間（毎週）の場合

```
day_of_the_week = 曜日の識別子,
hour = 時間,       // HH 形式
minute = 分,       // mm 形式
```

曜日の識別子には以下を指定。複数指定する場合はカンマで連結する。

表 5-10 曜日の識別子

識別子	説明
$EVENT.Mon	月曜日
$EVENT.Tue	火曜日
$EVENT.Wed	水曜日
$EVENT.Thu	木曜日
$EVENT.Fri	金曜日
$EVENT.Sat	土曜日
$EVENT.Sun	日曜日

時間（毎日）の場合

```
    hour = 時間,      // HH 形式
    minute = 分,      // mm 形式
```

時間（毎時）の場合

```
    interval = インターバル時間,      // 1～99
    minute = 分,      // mm 形式
```

時間（毎分）の場合

```
    interval = インターバル分,      // 1～120
```

ウィンドウ状態の場合

```
    window_rule_ref = Window_rule ウィンドウ識別名,
    win_state = 画面変化,
```

設定内容については、『8.3.4 ウィンドウ状態待機』を参照のこと。

メール受信の場合

設定値なし。

■ 呼び出し処理毎パラメータ

呼び出し処理によって設定値が異なる箇所について記載。

サブルーチン呼び出しの場合

```
    callsub 文字列または識別子 プリアンブル（ 式の並び ）,
```

設定内容については、『6.12 サブルーチン呼び出し文』を参照のこと。

シナリオファイル呼び出しの場合

```
    call_scenario プリアンブル（
      file = 変数名 または ファイル名,
      call_vars = ( 呼び出し先シナリオの変数名 1 = 初期値 1, ...),
      return_vars = ( 呼び出し先シナリオから値を引き継ぐ変数名 1, ... ),
      return_value = 結果返却先変数名
    ),
```

設定内容については、『6.13 シナリオファイル呼び出し文』を参照のこと

■ 説明

イベント監視名は、文字列名および識別子で記述する。
イベントトリガー条件、イベントトリガー条件毎パラメータ、呼び出し処理毎パラメータ、初期監視有無を指定する。
イベント監視名の重複はエラーとなる。
イベントトリガー条件とイベントトリガー条件毎パラメータの組み合わせが異なる場合はエラーとなる。

■ イベント監視部の例

```
Events = (
    "イベント監視名" [comment = "コメント"] = (
      trigger = $EVENT.UpdateFile,
      path = @"C:¥temp¥WinActor¥監視対象データ.txt",
      '返り値' = callsub "サブルーチングループ" [name = "イベント一覧:イベント監視名"] (),
      from_the_start = false
    )
)
```

5.9　ブレークポイント情報部

■ 構文

```
Breakpoint_info = (
   ( id = 定数式, enable = true または false ), ...
)
```

■ 説明

複数のブレークポイント情報を記載することができる。
ブレークポイント情報には ID 、および、ブレークポイントの有効または無効を指定する。
ID は、ブレークポイントを設定したノードのプリアンブルの ID 属性にて指定した ID 番号である。

■　ブレークポイント情報部の例

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

5.10　シナリオ情報部

■　構文

```
Scenario_info = (
    creator    = 作成者文字列,
    contact    = 連絡先文字列,
    expiration = 有効期限文字列,
    remarks    = 備考文字列,
    dataupdate_change = true または false,
    ignore_datawrite_error = true または false,
                                  // true のときデーター覧書き込み時のエラーを無視する
    variable_limit = true または false ,          // true のとき変数値の文字数を制限する
    save_ignore_exec = true または  false,        // true のとき実行抑止状態を保存する
    user_dictionary_enable = true または false,
                                 // true のときシナリオ実行時にユーザ変換辞書を利用する
    wss_integer_arithmetic_only = true または false,
     // true のときシナリオ内の式と定数式にあらわれる四則演算子をすべて整数演算として扱う
    wait_setting = タイムアウト設定,
    wait_timeout = タイムアウト時間,    // ミリ秒
    percent_variable = true または false,
    use_webdriver = true または false,
    collect_healing_info = true または false,
    alter_property_path = true または false
)
```

■　説明

WinActor のシナリオ情報画面にあたる部。
上記の構文で指定可能な属性については、WSS でシナリオ情報の値を変更可能である。
wss_integer_arithmetic_only は WSS で記述したシナリオでのみ使用できる。WinActor 本体のシナリオ情報には存在しない。
wait_setting にはタイムアウト時間の取得元を指定する。取得元には次のいずれかを指定する。

表 5-11 タイムアウト時間の取得元

取得元	説明
$SCENARIO_INFO.WaitSettingOption	オプション画面の指定
$SCENARIO_INFO.WaitSettingScenario	シナリオ情報の指定

wait_setting は省略可能。省略した場合は $SCENARIO_INFO.WaitSettingOption を指定したとみなす。
wait_timeout にはミリ秒単位のタイムアウト値を指定する。タイムアウト値は 100 から 3,600,000 の範囲にあること。この範囲を外れた場合は警告し、100 または 3,600,000 の近いほうの値が指定されたとみなす。
wait_timeout は省略可能。省略した場合は 10,000 が指定されたとみなす。
wait_setting に $SCENARIO_INFO.WaitSettingOption が指定されたときには wait_timeout の指定は使われない。
percent_variable には true または false を指定する。指定を省略したときには false を指定したとみなす。

use_webdriver には true または false を指定する。指定を省略したときには、WinActor Ver.7.4 より前（Ver.7.3.1 以前）で作成されたスクリプトの場合は true を指定したとみなす。WinActor Ver.7.4 以降で作成されたスクリプトの場合は false を指定したとみなす。

collect_healing_info には true または false を指定する。指定を省略したときには、false を指定したとみなす。

alter_property_path には true または false を指定する。指定を省略したときには、false を指定したとみなす。collect_healing_info = true の場合のみ意味をもつ。

■ シナリオ情報部の例

```
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 イメージ部

■ 構文

```
Images = (
  イメージ宣言 ,...
)
```

■ 説明

WinActor が保持しているイメージ画像が列挙される。
イメージ画像は WinActor で取得する。イメージ画像を WSS にだけ記述しても、WinActor に認識されない。

イメージ宣言 の イメージ ID が重複したらエラーとなる。

5.11.1 イメージ宣言

■ 構文

```
イメージ ID = ( 属性 = 属性値 , ...)
```

■ 属性

表 5-12 イメージ宣言の属性

属性	説明
name	イメージの名称文字列
size	イメージをキャプチャしたときの縮小率（パーセント）、0～100 の数値
width	イメージの幅サイズ数値
height	イメージの高さサイズ数値

5.11.2 イメージ宣言の例

```
Images = (
    img_20190613104343897 = (name = "無題-メモ帳", size = 50, width = 409, height = 194),
    img_20191210154248256 = (name = "start-Google 検索-InternetExplorer", size = 50, width = 562, height = 531)
)
```

5.12 フローチャート情報部

■ 構文

```
Flow_divide_info = (
    タブ名文字列 = ( seq = 定数式, tab_id = 文字列 ) , ...
)
```

■ 説明

タブ名文字列 が重複した場合はエラーとなる。

tab_id は重複しないこと。

tab_id = "0" は必須である。

tab_id は、フローティング部またはサブルーチン部のプリアンブルの tab_id_ref から参照される。tab_id_ref の参照先が見つからない場合はエラーとなる。

■ フローチャート情報部の例

```
Flow_divide_info = (
    "メイン" = (seq = 1, tab_id = "0"),
    "新規タブ_1" = (seq = 2, tab_id = "1")
)
```

5.13 単語辞書部

■ 構文

```
Translation = (
   (国識別文字列, 国識別文字列, ... ),
   (単語文字列,  単語文字列, ...),
   ...
)
```

■ 説明

各国語の単語対応辞書を設定する。
単語対応辞書の設定は、WinActor 本体でのみ可能である。WSS で記述した単語辞書部は、WinActor 本体に反映されない。

最初に、タイトルとして言語種別文字列の定数タプルを記述する。
引き続いて、対応する単語を並べた定数タプルを複数記述する。
すべてのタプルの個数は同一でなければならない。

■ 単語辞書部の例

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

6.　文（statement）

6.1　説明

英単語 ．，；（ ）｛ ｝は構文要素。日本語での記述部分は別に説明がある。
文のプリアンブルは省略可能
「文の並び」には、文を複数個記述可能。0 個でも構わない。
式 ； は文にならない。
空文も存在しないので、；だけは文としては許されない。

6.2　グループ文（group statement）

■　構文

```
Group プリアンブル
{
   文の並び
}
```

文の並びをまとめるのに使用する。
フローティング部に文の並びを記述したときには、コンパイラがグループ文にまとめる。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
プリアンブルは省略可能。

6.3　if 文

■　構文

```
if（ 条件式 ） プリアンブル
then プリアンブル
{
   文の並び
}
else プリアンブル
{
   文の並び
}
```

else 部の「else プリアンブル｛｝」は省略可能
then 部および else 部のプリアンブルは、name 属性のみが有効である。

ノードの付箋を表示したくないときには、各ノードのプリアンブルに TagVisible = false を指定する。

6.4 while 文

■ 構文

```
while プリアンブル
ループ条件
( Counter 識別子 )
{
  文の並び
}
```

カウンタ変数を指定する「(Counter 識別子)」は省略可能。

カウンタに特殊識別子は指定できない。定数宣言された変数も指定できない。

カウンタに無名識別子 '' を指定することは可能。

wihle Counter の 大文字小文字の差違は無視する。

プリアンブルで属性 isclosed_body を true に設定すると、while ノードのボディ部分が閉じて表示される。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

6.4.1 ループ条件

■ 構文

下記の 8 形式のいずれか。
詳しくは『WinActor 操作マニュアル』の『繰り返し』ノードを参照のこと。

表 6-1 ループ条件の形式

形式	説明
(条件式)	条件式は後述。条件式が偽になるまでループを回る。
(true)	常に真の条件式を表す。ループを回り続ける。
(false)	常に偽の条件式を表す。ループにはいることはない。
(Start = 式, End = 式)	指定された数値の範囲でループを回る。
(File = 文字列 または 変数名)	Excel または csv ファイルを指定してデータ数分ループを回る。

形式	説明
(DBSource = 文字列 または 変数名 , User = 文字列 または 変数名 , Password = 文字列 または 変数名 , Table = 文字列 または 変数名)	DB を使ってデータ数分ループを回る。
(Template_And_Data = 変数名, Iterate = true または false, IterateOver = 式, IsUpdate = true または false, UpdateTo = 変数名)	雛形・データを使用してデータ数分繰り返す。 IterateOver は Iterate = true の時のみ、 UpdateTo は IsUpdate = true の時のみ意味をもつ。
(Json_Object = 変数名 または JSON 配列文字列, Key = 式, KeyOut = 変数名, ValueOut = 変数名)	JSON オブジェクトまたは JSON 配列から、キーや値を取得して繰り返す。

while 文と dowhile 文で共通のループ条件構文。

丸かっこで囲む。

true false Start End File DBSource User Password Table

Template_And_Data Iterate IteratOver IsUpdate UpdateTo

Json_Object Key KeyOut ValueOut は大文字小文字を区別しない。

6.5 dowhile 文

■ 構文

```
dowhile プリアンブル
ループ条件
( Counter 識別子 )
{
  文の並び
}
```

カウンタ変数を指定する「(Counter 識別子)」は省略可能。

カウンタに特殊識別子は指定できない。定数宣言された変数も指定できない。

カウンタに無名識別子 '' を指定することは可能。

dowihle Counter の 大文字小文字の差違は無視する。

プリアンブルで属性 isclosed_body を true に設定すると、dowhile ノードのボディ部分が閉じて表示される。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

6.5.1　ループ条件

『6.4.1 ループ条件』に同じ。

6.6　switch 文

■　構文

```
switch プリアンブル
ケース文の並び
デフォルト文
```

デフォルト文は省略可能。

条件式に計算を伴う式を与えた場合、コンパイラは switch を if-then-else の条件分岐に置き換えるので、フローチャート上での表現は変化する。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

6.6.1　ケース文

■　構文

```
Case（ 条件式 ） プリアンブル
{
   文の並び
}
```

条件式は必須。

Case の大文字小文字の差違は無視する。

プリアンブルは省略可能。

プリアンブルは、name 属性のみが有効である。

6.6.2　デフォルト文

■　構文

```
Default
{
   文の並び
}
```

デフォルト文にはプリアンブルを付与できない。
Default の大文字小文字の差違は無視する。

6.7 try 文

■ 構文

```
try プリアンブル
{
   文の並び
}
キャッチ文の並び
```

キャッチ文は 1 個以上が必須
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
プリアンブルは省略可能。

6.7.1 キャッチ文

■ 構文

```
catch 例外処理名文字列 プリアンブル
{
   文の並び
}
```

プリアンブルは省略可能。有効な属性はない。

6.8 return 文

■ 構文

```
return（ 式 ） プリアンブル ;
```

return 文はサブルーチン部で定義するサブルーチンのブロック内でしか使えない。
「式」は省略できるが、(と) は省略不可。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
プリアンブルは省略可能。

6.9　シナリオ終了文

■　構文

```
scenario_return(式) プリアンブル ;
```

式、プリアンブルは省略可能。
式の結果は、呼び出し元シナリオの call_scenario への戻り値となる。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
メインルーチンに配置しない場合は警告が表示される。

6.10　break 文

■　構文

```
break プリアンブル ;
```

break 文は while または dowhile のブロック内でしか使えない。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
プリアンブルは省略可能。

6.11　continue 文

■　構文

```
continue プリアンブル ;
```

continue 文は while または dowhile のブロック内でしか使えない。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。
プリアンブルは省略可能。

6.12　サブルーチン呼び出し文

■　構文

```
callsub 文字列または識別子 プリアンブル （ 式の並び ）;
```

「文字列または識別子」に対応するサブルーチンを定義すること。
callsub は省略可能

「式の並び」は省略できるが、(と) は省略不可。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

6.13 シナリオファイル呼び出し文

■ 構文

```
call_scenario プリアンブル (
  file =   変数名 または ファイル名,
  call_vars = ( 呼び出し先シナリオの変数名 1 = 初期値 1, ...),
  return_vars = ( 呼び出し先シナリオから値を引き継ぐ変数名 1, ... ),
  return_value = 結果返却先変数名
) ;
```

call_vars には、呼び出し先シナリオの変数名とその初期値のペアを設定する。ペアは省略可能。

呼び出し先シナリオの変数名 が呼び出し先のシナリオにない場合は無視される。

初期値を与えない場合は無名識別子 '' を置く。

名前を指定しないで値だけを設定する場合は、 無名識別子=値, とする。この場合、スクリプトを読み込み時に警告が表示される。

return_vars には、呼び出し先シナリオが終了したときに値を引き継ぐ変数名を指定する。変数名は省略可能。

return_value で指定した変数に、呼び出し先シナリオからの戻り値が格納される。

戻り値は、呼び出し先シナリオの scenario_return 文で与えられる。

代入や式の中に現れた場合は、結果返却先変数名に戻り値は格納されない。

結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

例

```
    call_scenario [name = "シナリオファイル呼び出し", comment = ""]
    (
        file = @"C:¥Users¥user¥Documents¥sub.ums7",
        call_vars = (name = "user"),
```

```
                    //シナリオファイル sub.ums7 の変数 name に "user" を設定して起動する
    return_vars = (ret)          //シナリオファイル sub.ums7 の変数 ret の値を引き継ぐ
);


result = call_scenario [name = "シナリオファイル呼び出し", comment = ""]
                                        // scenario_return で設定された結果を代入
(
    file = @"C:¥Users¥user¥Documents¥sub2.ums7",
    call_vars = (),       // 変数と値を省略
    return_vars = ()      // 変数を省略
);
```

6.14 アダプタアクション文

■ 構文

```
アダプタアクション ;
```

アダプタアクションを呼び出す。

値を返すアクションの場合、アクション引数で値の返し先を設定しないと値は捨てられる。

値を返すアクションを呼び出す場合、代入文の右辺にアダプタアクションを置くことを推奨する。フローチャートの WSS 出力で代入文となったときは返し先の設定は省略される。

アダプタアクションの章を参照のこと。

6.15 代入文

■ 構文

```
識別子 = 式 プリアンブル ;
```

式の値を変数に代入する。

= の左辺におく識別子は次のいずれか。

表 6-2 代入文での識別子

識別子
変数部で変数として宣言されている識別子
特殊識別子で読み書き可能な識別子

代入文のプリアンブルと代入文の右辺にある要素につけたプリアンブル（アダプタアクションなど）の両方に（ブレークポイント用などの）ID を指定した場合、両方の ID を有効とするために、コンパイラの代入操作削除の最適化はなされない。

右辺の要素のプリアンブルにだけ ID を指定すると、コンパイラ生成ノードを減らせる可能性がある。

ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは省略可能。

6.16 四則演算

四則演算は自由な記述が可能。

コンパイラがノードに変換する際に、必要なノードを追加するため、フローチャート上の表現が変化することがある。

7.　式（expression）

1 つ以上の因子を四則演算子 ＋ － ＊ ／　+^　−^　*^　/^ で連結したもの。

因子の前に単項演算子の ＋ －　+^　−^ を置いてよい。

演算子の優先順位は高い順に次のとおり。

表 7-1 演算子の優先順位

No.	演算子
❶	単項演算子の ＋　－　+^　−^
❷	＊　／　*^　/^
❸	＋　－　+^　−^

実行時の因子が文字列であった場合は、単項演算子と四則演算子による演算はできない。ただし、数値として解釈できる文字列は演算できる。

論理値を得る演算はない。

+^　−^　*^　/^ は整数演算に使用する。フローチャートの四則演算ノードの「整数として計算し、計算結果の小数点以下を切捨てる」のチェックを付けた場合に相当する。

通常の ＋ － ＊ ／ と混合して使用してよい。

被演算子の値が整数でなかった場合は実行時エラーとなる。カンマ桁区切りの数値表記も整数ではないため実行時エラーとなる。

単項演算子の +^ と −^ を因子の前に置いたときには、それぞれ 「(0 +^ 因子)」「(0 −^ 因子)」の式として扱い、整数かどうかのチェックがなされる。

「−^ −^ 因子」のように単項の整数減算演算子を偶数個置いたときには、(0 +^ 因子）の意味となる。

因子が全角の数字やカンマ桁区切りの数値表記であっても、演算結果は半角の数字となる。

7.1　因子（factor)

■　構文

次のいずれか。

表 7-2 因子の構文

因子	備考
整数	
浮動小数	

因子	備考
文字列	単項演算 四則演算不可
識別子	
TRUE	文字列 "TRUE" と解釈される。 単項演算 四則演算不可。
FALSE	文字列 "FALSE" と解釈される。 単項演算 四則演算不可。
戻り値のあるアダプタアクション	引数リストの返し先の設定は無視される。 WSS 出力では返し先設定が省略される。
戻り値のあるサブルーチン呼び出し	
戻り値のあるシナリオファイル呼び出し	
(式)	

識別子は変数、定数、既定義定数、特殊変数、無名識別子のいずれか。

無名識別子と値が文字列となる識別子は単項演算と四則演算ができない。

7.2 定数式

コンパイル時に演算して定数を得る式。論理値を得る演算もある。

1 つ以上の定数因子を定数式用二項演算子で連結したもの。

数値として解釈できる文字列も数値として演算できる。ただし、比較演算子の両辺とも文字列であった場合は、数値として解釈されない。

&& (かつ) と || (または) の引数は両辺とも論理値であること。また、両辺とも評価される。

~ (正規表現マッチ) と !~ (正規表現アンマッチ) の引数は両辺とも文字列であること。

* / + - の引数は両辺とも数値、または数値と解釈できる文字列であること。

*^ /^ +^ -^ の引数は両辺とも整数値であること。小数点以下の値を持つ場合はエラーとなる。カンマ桁区切りの数値表記は整数ではないためエラーとなる。

== != >= > < <= の引数は両辺とも数値、または、両辺とも文字列であること。

数値と文字列の演算はできない。

両辺とも文字列であった場合、数値として解釈できる文字列であっても数値として解釈されない。例えば "2000" < "300" は両辺とも文字列として演算されて true となる。"2000" < 300 は両辺とも数値と解釈されて false となる。

両辺が文字列であった場合、 == と != は大文字小文字を区別した完全一致の比較となる。

>= > < <= は大文字小文字を無視した辞書順比較となる。

7.2.1 定数式用二項演算子

優先順位は高い順に次のとおり。

表 7-3 定数式用二項演算子

No.	二項演算子	備考
❶	＊　／　＊^　／^　&&	
❷	＋　－　+^　−^　||	定数因子の前に、単項演算子の ＋ － +^ −^ を置くこともできる。
❸	==　!=　>=　>　<　<=　~　!~	

7.2.2 定数因子

■ 構文

次のいずれか。

表 7-4 定数因子の構文

定数因子	備考
整数	
浮動小数	
文字列	単項演算不可。
識別子	
TRUE	文字列 "TRUE" と解釈される。 ＋ － +^ −^ の単項演算不可。
FALSE	文字列 "FALSE" と解釈される。 ＋ － +^ −^ の単項演算不可。
strcmp(定数式, 定数式)	定数式は文字列であること。 大文字小文字全角半角の区別ありの比較を実施し、結果は "true" または "false"。
strcasecmp(定数式, 定数式)	定数式は文字列であること。 大文字小文字全角半角の区別なしの比較を実施し、結果は "true" または "false"。
! 定数因子	定数因子は論理値であること。
(定数式)	

識別子は定数、既定義定数、読み込み専用特殊変数、無名識別子のいずれか。

無名識別子は演算ができない。

未定義の識別子は、警告を出力し、値を整数のゼロとみなして処理は続行する。

7.3 条件式

1 つ以上の条件式因子を && または || で連結したもの。

&& を優先するが、できるだけ括弧で囲むこと推奨する。

条件式の結果は "true" または "false" になる。

7.3.1 条件式用二項演算子

次のいずれか。

表 7-5 条件式用二項演算子

二項演算子
== != >= > < <= ~ !~

7.3.2 条件式因子

条件式因子の結果は "true" または "false" になる。

■ 構文

次のいずれか。

表 7-6 条件式因子の構文

条件式因子	備考
istrue(式)	結果は "true" または "false"。
isfalse(式)	結果は "true" または "false"。
strcmp(式, 式)	式は文字列であること。 大文字小文字全角半角の区別ありの比較を実施。
strcasecmp(式, 式)	式は文字列であること。 大文字小文字全角半角の区別なしの比較を実施。
式 条件式用二項演算子 式	
TRUE	
FALSE	
! 条件式因子	
(条件式)	

8.　アダプタアクション

アクションカテゴリのノードに対応する。

値を返すアダプタアクションは式に含めることができる。

アダプタ引数リストの一般記法は構造体を参照のこと。

■　構文

```
WinActor.アクション名 プリアンブル アダプタ引数リスト
```

アクション名の大文字小文字の差違は無視する。

値を返すアクションは、特定の属性名（value など）に、値の返し先変数名を指定する。
値を返すアクションを式（代入文含む）に使用した場合、属性名で指定した返し先は無視する。この場合、返し先の属性名は省略可能。またフローチャートの WSS 出力で式（代入文）になったときには返し先属性名による指定は省略される。
ノードの付箋を表示したくないときには、プリアンブルに TagVisible = false を指定する。

プリアンブルは一部を除いて省略可能。

以下、アクションごとにアダプタ引数リストを示す。

8.1　自動記録アクション

8.1.1　イベント記録 クリック

ボタン、チェックボックス、ラジオボタンをクリックする操作

```
WinActor.ClickWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

window_rule_ref, control, capture は必須。

Window_rule ウィンドウ識別名には「ウィンドウマッチルール部」の「ウィンドウ識別名」を記述する。(以下同様)

control の属性に付属している <チェック> は、イベントモードで記録されたノードのプロパティに表示される詳細設定タブのチェックボックスの指定となる。

<true> はチェックあり、<false> はチェックなしにあたる。

control の instance には、コントロールに割り当てられた通番を数値、または数値が格納された変数を指定。(以下同様)

control の text には、コントロールに表示されている文字列を文字列、または文字列が格納された変数を指定。(以下同様)

control の「ポジション」は変数、文字列、または、(x＝ 定数式, y＝ 定数式) とする。(以下同様)

指定しないときには文字列を "" とする。あるいは 2 つの定数式を "" とする。(x＝"", y＝"") のように記述する。

変数または文字列は位置の座標数値をカンマで区切って指定する。例 100,200 。

表 8-1 タイムアウト時間の取得元

取得元	説明
$WIN32.WaitSettingOption	オプション画面の指定
$WIN32.WaitSettingScenario	シナリオ情報の指定
$WIN32.WaitSettingNode	wait_timeout で指定

wait_setting は省略可能。省略した場合は $WIN32.WaitSettingScenario を指定したとみなす。

wait_timeout にはミリ秒単位のタイムアウト値またはタイムアウト値を保持した変数名を指定する。タイムアウト値は 100 から 3,600,000 の範囲にあること。この範囲を外れた場合は警告し、100 または 3,600,000 の近いほうの値が指定されたとみなす。

変数名として無名識別子 '' （シングルクオート 2 個）が指定されたときには 10,000 が指定されたとみなす。

wait_timeout は省略可能。省略した場合は 10,000 が指定されたとみなす。

wait_setting に $WIN32.WaitSettingOption または $WIN32.WaitSettingScenario が指定されたときには wait_timeout の指定は使われない。また、WSS ファイルにはシナリオのノードプロパティのタイムアウト設定を「ノードで指定」としたときのみ出力される。

capture の imageid には、イメージ部のイメージ宣言にあるイメージ ID を指定する。

capture のマウス操作位置X マウス操作位置Y は数値定数。指定しない場合は "" とする。

8.1.2 イベント記録 文字列設定

テキストボックスに文字列を設定する操作

```
WinActor.SetTextWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポ
ジション),
    value           = 式,    // 設定するテキスト文字列
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

value に設定する式の結果が数値の場合は、表記を文字列とみなして出力する。

設定しないときには value に無名識別子 '' （シングルクオート 2 個）を指定する。

"" （ダブルクオート 2 個）を指定すると空文字列を指定したことになる。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.3 イベント記録 リスト選択

リストボックスやコンボボックスの項目を選択する操作

```
WinActor.SelectListWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポ
ジション),
    value           = 式、または、変数名,
    kind            = 選択種別,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

選択種別 kind には $SelectListWin32.index または $SelectListWin32.text を指定する。

表 8-2 リスト選択アクションの選択種別

選択種別	説明
$SelectListWin32.index	ゼロ始まりの番号を指定してリストを選択
$SelectListWin32.text	リストに表示されている文字列でリストを選択

value にはリストボックスを選択する値を得る式、もしくは値が格納された変数名を指定する。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.4 イベント記録 タブ選択

タブの切り替え操作

```
WinActor.SelectTabWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    value           = 式、または、変数名
    kind            = 選択種別,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

選択種別 kind には $SelectTabWin32.index または $SelectTabWin32.text を指定する。

表 8-3 タブ選択の選択種別

選択種別	説明
$SelectTabWin32.index	ゼロ始まりの番号を指定してタブを選択
$SelectTabWin32.text	タブに表示されている文字列でタブを選択

value には タブを選択する値を得る式、もしくは値が格納された変数名を指定する。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.5 イベント記録 エミュレーション

マウスをクリックした位置やキーボードを操作した順序を覚えて自動に操作。

```
WinActor.EmulationWin32 プリアンブル
```

```
(
  window_rule_ref = Window_rule ウィンドウ識別名,
  action          = アクション列,
  wait_setting    = タイムアウト設定,
  wait_timeout    = タイムアウト時間,    // ミリ秒
  capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
)
```

値は返さない。

action にはマウス操作、キーボード操作、待ちを複数記述する。記述の順番に実行される。

アクション列は、次のいずれか。

表 8-4 エミュレーションのアクション列

アクション列	説明
アクション	アクションを 1 つだけ記述する場合
(アクション, ...)	複数のアクションを記述する場合は、カンマで区切り、丸かっこで囲む。

アクションは、次の 3 種類のアクションを記述可能。まぜてもよい。

表 8-5 エミュレーションのアクション

アクション
マウスアクション
キーアクション
待ち

マウスアクション

```
@(Mouse, ボタン, 動作, X 位置, Y 位置, 基点, X_D/P, Y_D/P, Scale)
```

表 8-6 エミュレーションのマウスアクション

マウスアクション	説明	
ボタン	次のいずれか。	
	L	左ボタン
	R	右ボタン
	M	中央ボタン
動作	次のいずれか。	
	DOWN	ボタンのダウン
	UP	ボタンのアップ
	MOVE	ムーブ

マウスアクション	説明	
	DBL	ボタンのダブルクリック
基点	次のいずれか。(大文字小文字の差異は問わない)	
	LEFTTOP	左上基点
	RIGHTTOP	右上基点
	LEFTBOTTOM	左下基点
	RIGHTBOTTOM	右下基点
X_D/P Y_D/P	X 位置、Y 位置の指定方法を指定する。 次のいずれか。(大文字小文字の差異は問わない)	
	D	座標直接 (Direct)
	P	パーセント
Scale	キャプチャイメージの拡大率	1.0 が等倍 省略可能

キーアクション

```
@(Key, キーコード, UP/DOWN)
```

待ち

```
@(Wait, 待ち時間)
```

待ち時間の単位はミリ秒

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

例

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

// マウス操作と待ち
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)),

// キーボード操作と待ち
action = (@(Key, 18, DOWN),
          @(Key, 115, DOWN),
          @(Key, 115, UP),
          @(Key, 18, UP),
          @(Wait, (DefaultWaitTime + 2 * 500))),     // 定数式はかっこで囲む
```

```
// アクションを 1 つだけ記述
action = @(Wait, 300),
```

8.1.6 イベント記録 文字列取得

```
WinActor.GetTextWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポ
ジション),
    value           = 結果返却先変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,   // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

結果を value で指定した変数に返す。書き込み可能な変数を指定すること。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

例

```
Var_group (
  var01 = 0 [comment = "作業変数"],
  ret01 = 0 [comment = "返却用"]
)
// 変数に結果を代入する。
ret01 = WinActor.GetTextWin32 [name = "文字列取得 (WIN32) ", comment = "get text"]
(
    window_rule_ref = "無題-メモ帳",
    control         = (instance<true> = var01, text<true> = var01, position<true> = var01),
    capture         = (imageid = "img_20191115153616376", x = 1197, y = 159)
);
// 結果を返す変数名を指定 (value 属性)
WinActor.GetTextWin32 [name = "文字列取得 (WIN32) ", comment = "get text"]
(
    window_rule_ref = "無題-メモ帳",
    control         = (instance<true> = var01, text<true> = var01, position<true> = var01),
    value           = ret01,
    capture         = (imageid = "img_20191115153616376", x = 1197, y = 159)
```

```
  );
```

8.1.7 イベント記録 リスト取得

```
WinActor.GetListWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    value           = 結果返却先変数名,
    kind            = 選択種別,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

選択種別 kind には $GetListWin32.index または $GetListWin32.text を指定する。

表 8-7 リスト取得の選択種別

選択種別	説明
$GetListWin32.index	リストで選択されている要素のインデックスを取得
$GetListWin32.text	リストで選択されている要素の名前を取得

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.8 イベント記録 チェック状態取得

```
WinActor.GetCheckWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    value           = 結果返却先変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
```

```
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

true または false の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.9 イベント記録 有効無効状態取得

```
WinActor.GetEnableWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    value           = 結果返却先変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

true または false の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.10 イベント記録 リスト一括取得

```
WinActor.GetAllListWin32 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    file            = 変数名 または ファイル名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture       = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

file に結果を出力するファイル名文字列、またはファイル名を格納した変数名を指定する。

wait_setting と wait_timeout については『8.1.1 イベント記録 クリック』の説明を参照のこと。

8.1.11 UIAutomation

```
WinActor.UIAutomation プリアンブル
  (
      window_rule_ref = Window_rule ウィンドウ識別名,
      path = コントロールパス JSON 形式文字列,
      expand_variable = true / false,    // 変数埋め込み指定
      pattern = コントロールパターン,
      action = アクション,
      use_wildcard = true / false,    //  “*” をつかった曖昧指定の有効化設定
      wait_setting = タイムアウト設定,
      wait_timeout = タイムアウト時間,  // ミリ秒
      wait_timeout_period = 待機条件,
      cache_update = true / false,       // キャッシュ強制更新フラグ デフォルト false
      wait_retry_max = リトライの上限回数,
      path_version = $UIA.Version2,      // または $UIA.Version1
      activate_target = true / false,
                                // 実行時に対象ウィンドウをアクティブにする デフォルト true
      control = コントロール指定,
      result = 結果返却先変数名,
      capture = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

path には、対象のコントロールを特定する コントロールパス を JSON 形式で表わし、文字列で記述する。

JSON 形式にはダブルクオート文字が頻繁に現れるので、文字列はブロック逐語的文字列で記述するとよい。

path は必須。

path 内の文字列で変数を展開するには、文字列内に %変数名% を記述し、expand_variable に true を与える。

指定のないときには false が指定されたとみなす。

pattern には、対象コントロールに対する操作である コントロールパターン を指定する。

pattern は省略可能。省略した場合は action で指定された アクション に対応するコントロールパターンが使用される。

コントロールパターンとアクションの有効な組み合わせは『表 8-11 UIAutomation のアダプタ引数リスト』を参照すること。

表中の識別子は共通の接頭辞 $UIA. を省略している。

action には、コントロールパターン毎に定められた操作である アクション を指定する。

action は必須。

コントロールパターンとアクションの有効な組み合わせは『表 8-11 UIAutomation のアダプタ引数リスト』を参照すること。

表中の識別子は共通の接頭辞 $UIA. を省略している。

use_wildcard には UIAutomation 対象コントロール指定に "*" を使った曖昧指定の有効・無効を指定する。

true を指定したときには曖昧指定を有効にする。

false を指定したときには曖昧指定を無効にする。

use_wildcar は省略可能。省略したときには false（曖昧指定無効）が指定されたとみなす。

wait_setting には、UI オートメーションノード実行時に、対象コントロールを含むウィンドウおよび対象コントロール自体が存在する状態になるまでのタイムアウト時間の取得元を指定する。取得元には次のいずれかを指定する。

表 8-8 タイムアウト時間の取得元

取得元	説明
$UIA.WaitSettingOption	オプション画面の指定
$UIA.WaitSettingScenario	シナリオ情報の指定
$UIA.WaitSettingNode	wait_timeout で指定

wait_setting は省略可能。省略した場合は $UIA.WaitSettingNode を指定したとみなす。

wait_timeout にはミリ秒単位のタイムアウト値またはタイムアウト値を保持した変数名を指定する。タイムアウト値は 100 から 3,600,000 の範囲にあること。この範囲を外れた場合は警告し、100 または 3,600,000 の近いほうの値が指定されたとみなす。

変数名として無名識別子 '' （シングルクオート 2 個）が指定されたときには 30,000 が指定されたとみなす。

wait_timeout は省略可能。省略した場合は 30,000 が指定されたとみなす。

wait_setting に $UIA.WaitSettingOption または $UIA.WaitSettingScenario が指定されたときには wait_timeout の指定は使われない。

wait_timeout_period には待機の条件を指定する。待機条件には次のいずれかを指定する。

表 8-9 待機条件

待機条件	説明
$UIA.WaitForWindow	ウィンドウが見つかるまで待機
$UIA.WaitForControl	対象コントロールが見つかるまで待機

wait_timeout_period は省略可能。省略した場合は $UIA.WaitForControl が指定されたとみなす。

cache_update には UI オートメーションノード実行に使用しているキャッシュの強制更新有無を指定する。

true を指定したときには強制的に更新する低速モードとなる。

false を指定したときには必要な場合のみ更新する高速モードとなる。

cache_update は省略可能。省略した場合は false（高速モード）を指定されたとみなす。

wait_retry_max には許可されない操作の実行エラーが発生した時に再度実行する上限回数を 10 進数で指定する。

0 はリトライしないことを意味する。

負数を指定したときには回数の上限なしを意味する。

wait_retry_max は省略可能。省略した場合は 5 が指定されたとみなす。

path_version にはコントロールパス書式のバージョンを指定する。省略した場合は $UIA.Version1 が指定されたとみなす。

表 8-10 コントロールパス書式のバージョン

書式バージョン	説明
$UIA.Version1	バージョン 1（WinActor7.4.4 以前と互換）の書式を指定する
$UIA.Version2	バージョン 2 の書式を指定する

activate_target には実行時に対象ウィンドウをアクティブにするかを指定する。

true を指定したときには、実行時に対象ウィンドウをアクティブにしてから対象コントロールを操作する。

false を指定したときには、対象ウィンドウをアクティブにせずに対象コントロールを操作する。

active_target は省略可能。省略した場合は true が指定されたとみなす。

control には、アクションに必要な パラメータ をアダプタ引数リストの形式で指定する。

control は省略可能。

パラメータは scroll, selection, value の 3 つのうちの 1 つを指定する。すべてのパラメータは下記のとおり。

各アクションで有効なパラメータは『表 8-11 UIAutomation のアダプタ引数リスト』を参照すること。

有効でないパラメータは、ファイルを読み込む際にメッセージを表示して、無視する。

```
control = (scroll    = (hscroll_amount = スクロール方向量識別子,
                        vscroll_amount = スクロール方向量識別子),
           selection = (item_index = 式, item_value = 式),
           value     = (item_value = 式, mode = モード指定識別子)
          )
```

アクションによって得られた結果を result で指定した変数に格納する。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

例

```
WinActor.UIAutomation [name = "選択・値で取得・値", 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 = "共通・ラベル取得", comment = ""]
    (
        window_rule_ref = "タイトル_*無題-メモ帳",
        path = @"""
[{"id":"Item 5","index":"0"}]
""",
```

```
        pattern = $UIA.CommonPattern,
        action = $UIA.GetName,
        control = (),
        capture = (imageid = "img_20200929092507503", x = 445, y = 10)
    );
```

表 8-11 UIAutomation のアダプタ引数リスト

pattern コントロール パターン $UIA.___	action アクション $UIA.___	control パラメータ						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							○

pattern コントロール パターン $UIA.___	action アクション $UIA.___	control パラメータ						result
		scroll		selection		value		
		hscroll_amount	vscroll_amount	item_index	item_value	item_value	mode	
	GetSelectableItems							○
	SelectItemByText				○2			
	SelectItemByIndex			○2				
SelectionItemPattern	IsSelected							○
	SelectAdditionally							
	Unselect							
	SelectOne							
TogglePattern	Toggle							
	GetToggleState							○
ValuePattern	IsReadOnly							○
	GetValue							○
	SetValue					○3	○4	
UnknownPattern	Unknown							

○1 省略時 $UIA.NoAmount 、 ○2 省略時 0、 ○3 省略時 0、

○4 省略時 $UIA.ModeNormal 、

コントロールパターンとアクションの識別子には $UIA. を付与する

8.1.12 UIAutomation ライブラリ

表 8-12 に示すアダプタアクションは UIAutomation でよく使われる操作について、コントロールパターン（pattern）とアクション（action）が設定済となっている。

機能は設定された組み合わせでの UIAutomation アダプタアクションと同じである。

```
WinActor.アダプタアクション名 プリアンブル
  (
      window_rule_ref = Window_rule ウィンドウ識別名,
      path = コントロールパス JSON 形式文字列,
      expand_variable = true / false,    // 変数埋め込み指定
      use_wildcard = true / false,    // “*” をつかった曖昧指定の有効化設定
      wait_setting = タイムアウト設定,
      wait_timeout = タイムアウト時間,  // ミリ秒
      wait_timeout_period = 待機条件,
```

```
    cache_update = true / false,       // キャッシュ強制更新フラグ デフォルト false
    wait_retry_max = リトライの上限回数,
    path_version = $UIA.Version2,      // または $UIA.Version1
    control = コントロール指定,
    result = 結果返却先変数名,
    capture = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
)
```

アダプタアクション名ごとに設定されているコントロールパターンとアクションを表に示す。pattern と action は表の値で固定されており、指定できない。

表 8-12 UIAutomation ライブラリのコントロールパターンとアクション

アダプタアクション名	設定済コントロールパターン	設定済アクション
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

他のアダプタ引数については『8.1.11 UIAutomation』を参照のこと。

8.1.13 UIAutomation ダンプ

```
WinActor.UiaDump プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    output_filename = ファイル名 または 変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

output_filename にダンプ結果を出力するファイル名文字列、またはファイル名を格納した変数名を指定する。

表 8-13 タイムアウト時間の取得元

取得元	説明
$UIADUMP.WaitSettingOption	オプション画面の指定
$UIADUMP.WaitSettingScenario	シナリオ情報の指定
$UIADUMP.WaitSettingNode	wait_timeout で指定

wait_setting は省略可能。省略した場合は $UIADUMP.WaitSettingScenario を指定したとみなす。

wait_timeout にはミリ秒単位のタイムアウト値またはタイムアウト値を保持した変数名を指定する。タイムアウト値は 100 から 3,600,000 の範囲にあること。この範囲を外れた場合は警告し、100 または 3,600,000 の近いほうの値が指定されたとみなす。

変数名として無名識別子 '' （シングルクオート 2 個）が指定されたときには 10,000 が指定されたとみなす。

wait_timeout は省略可能。省略した場合は 10,000 が指定されたとみなす。

wait_setting に $UIADUMP.WaitSettingOption または $UIADUMP.WaitSettingScenario が指定されたときには wait_timeout の指定は使われない。また、WSS ファイルにはシナリオのノードプロパティのタイムアウト設定を「ノードで指定」としたときのみ出力される。

8.2 自動記録アクション（IE）

8.2.1 IE モード記録 クリック

```
WinActor.ClickIE8 プリアンブル
 (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
 )
```

値は返さない。

wait_setting にはシナリオ実行時の IE アクションのタイムアウト時間の取得元を指定する。取得元には次のいずれかを指定する。

表 8-14 タイムアウト時間の取得元

取得元	説明
$IE.WaitSettingOption	オプション画面の指定
$IE.WaitSettingScenario	シナリオ情報の指定
$IE.WaitSettingNode	wait_timeout で指定

wait_setting は省略可能。省略した場合は $IE.WaitSettingScenario を指定したとみなす。

wait_timeout にはミリ秒単位のタイムアウト値またはタイムアウト値を保持した変数名を指定する。タイムアウト値は 100 から 3,600,000 の範囲にあること。この範囲を外れた場合は警告し、100 または 3,600,000 の近いほうの値が指定されたとみなす。

変数名として無名識別子 '' （シングルクオート 2 個）が指定されたときには 10,000 が指定されたとみなす。

wait_timeout は省略可能。省略した場合は 10,000 が指定されたとみなす。

wait_setting に $IE.WaitSettingOption または $IE.WaitSettingScenario が指定されたときには wait_timeout の指定は使われない。また、WSS ファイルにはシナリオのノードプロパティのタイムアウト設定を「ノードで指定」としたときのみ出力される。

control には操作対象特定種別を指定する。frame index のように空白を含む名前はクオートで囲む。

control の種別に付属している <チェック> は、IE モードで記録されたノードのプロパティに表示される詳細設定タブのチェックボックスの指定となる。

<true> はチェックあり、<false> はチェックなしにあたる。

指定しない種別は省略可能。

種別は、次のとおり。IE モード記録の control 属性は以下同様。

表 8-15 クリックの種別

種別	説明
tag	html の要素名を値入力または変数で指定。値は文字列
'frame index'	ドキュメント内のフレームに割り当てられた通番を値入力または変数で指定。値は数値
'tag index'	フレーム内の要素に割り当てられた通番を値入力または変数で指定。値は数値
ie_control_name	tag に付けられた name 属性値を値入力または変数で指定。値は文字列
type	tag に付けられた type 属性値を値入力または変数で指定。値は文字列
id	tag に付けられた id 属性値を値入力または変数で指定。値は文字列
value	tag に付けられた value 属性値を値入力または変数で指定。値は文字列

8.2.2 IE モード記録 文字列設定

```
WinActor.SetTextIE8 プリアンブル
 (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    value           = 文字列 または 変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
 )
```

値は返さない。

value に設定する文字列、または、文字列を格納した変数名を指定する。

value 属性の定数は数値でも文字列に変換する。

設定しないときには value に無名識別子 ''（連続した 2 つのシングルクオート）を指定する。""（連続した 2 つのダブルクオート）を指定すると空文字列を指定したことになる。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.3 IE モード記録 リスト選択

```
WinActor.SelectListIE8 プリアンブル
 (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    value           = 式、または、変数名,
    kind            = 選択種別,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
 )
```

値は返さない。

選択種別 kind には $SelectListIE8.index または $SelectListIE8.text を指定する。

表 8-16 リスト選択の選択種別

選択種別	説明
$SelectListIE8.index	ゼロ始まりの番号を指定してリストを選択

選択種別	説明
$SelectListIE8.text	リストに表示されている文字列でリストを選択

value にはリストボックスを選択する値を得る式、もしくは値が格納された変数名を指定する。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.4 IE モード記録 文字列取得

```
WinActor.GetTextIE8 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    value           = 結果返却先変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.5 IE モード記録 リスト取得

```
WinActor.GetListIE8 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    value           = 結果返却先変数名,
    kind            = 選択種別,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

選択種別 kind には $GetListIE8.index または $GetListIE8.text を指定する。

表 8-17 リスト取得の選択種別

選択種別	説明
$GetListIE8.index	リストで選択されている要素のインデックスを取得
$GetListIE8.text	リストで選択されている要素の名前を取得

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.6 IE モード記録 チェック状態取得

```
WinActor.GetCheckIE8 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    value           = 結果返却先変数名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

true または false の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.7 IE モード記録 有効無効状態取得

```
WinActor.GetEnableIE8 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
```

```
    control          = (種別<チェック> = 値, ...),
    value            = 結果返却先変数名,
    wait_setting     = タイムアウト設定,
    wait_timeout     = タイムアウト時間,    // ミリ秒
    capture          = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
 )
```

true または false の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.8 IE モード記録 表の値取得

```
WinActor.GetTableinfoIE8 プリアンブル
 (
    window_rule_ref    = Window_rule ウィンドウ識別名,
    control            = (種別<チェック> = 値, ...),
    get_tableinfo_mode = 取得モード,
    valuerow           = 行番号指定,
    valuecolumn        = 列番号指定,
    result             = 結果返却先変数名,
    file               = ファイル名文字列、または、ファイル名を格納した変数名,
    wait_setting       = タイムアウト設定,
    wait_timeout       = タイムアウト時間,    // ミリ秒
    capture             = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作
位置 Y)
 )
```

取得モード get_tableinfo_mode には取得モードの次のいずれかを指定する。

表 8-18 表の値取得の取得モード

取得モード	説明
$IEGetTableInfo.GetCell	セルの値取得
$IEGetTableInfo.ExistCell	セルの存在チェック(true/false)
$IEGetTableInfo.GetRow	行数取得
$IEGetTableInfo.GetColumn	列数取得
$IEGetTableInfo.GetAll	表の一括取得

取得モードがセルの値取得またはセルの存在チェックの場合は、

行番号指定 valuerow に、行番号の数値、または行番号を格納した変数名を指定する。

列番号指定 valuecolumn に、列番号の数値、または列番号を格納した変数名を指定する。

取得モードに $IEGetTableInfo.GetAll を指定した場合には、結果を file で指定したファイルに CSV 形式で書き出す。

取得モードに $IEGetTableInfo.GetAll 以外を指定した場合には、結果を result で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.2.9 IE モード記録 リスト一括取得

```
WinActor.GetAllListIE8 プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (種別<チェック> = 値, ...),
    file            = 変数名 または ファイル名,
    wait_setting    = タイムアウト設定,
    wait_timeout    = タイムアウト時間,    // ミリ秒
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

値は返さない。

file に結果を出力するファイル名文字列または、ファイル名を格納した変数名を指定する。

wait_setting と wait_timeout については『8.2.1 IE モード記録 クリック』の説明を参照のこと。

8.3 アクション、ユーザ、変数

8.3.1 画像マッチング

```
WinActor.ImageMatch [_OriginalID_f63cb690ce1d = 画像 ID 番号]
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    targetrange     = (x = X 位置, y = Y 位置, width = 幅, height = 高さ), // 画像の探索部分
```

```
    rawtargetrange  = (x = X 位置, y = Y 位置, width = 幅, height = 高さ), // マウス部分矩形
範囲
    mousecoordinate = (enable = true / false, x = X 位置, y = Y 位置),
                                                        // マウスアクション操作位置
    mouseaction     = マウスアクション,
    scale           = 縮尺率,                  // 縮尺率
    selectshape     = 矩形・円形モード,        // 赤枠形式 矩形・円形モードの指定
    similarity      = マッチ率,           // 0-100 %
    timeout         = タイムアウト時間,     // ミリ秒
    searchrange     = (enable = true / false , x = X 位置, width = 幅, y = Y 位置, height =
高さ,
                      startpoint= 開始点, x_coordinate = 座標指定, y_coordinate = 座標指
定),
    realtime        = true / false,         // 実行時にマッチング画像を取得するとき true
    realtimefile    = ファイルパス文字列 または 変数名,
                          // realtime = true の場合に、ファイル名またはフォルダ名を指定する
    useredframe     = true / false,  // パス指定マッチング画像と赤枠指定マッチング画像の
                                    // 両方を使用するとき true
    pathspecified   = ファイル・フォルダ指定, // パス指定マッチング画像の指定方法
    imagesplit      = true / false,      // 分割マッチングの場合 true
    value           = 結果返却先変数名,
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

プリアンブルの _OriginalID_f63cb690ce1d 属性は WinActor で保持している画像を識別する ID を指定しているため、変更しないこと。

プリアンブルには、comment や name を追加可能。

画像ファイルは、WinActor のシナリオ編集画面で編集（変更、追加など）する必要がある。

WSS で名前を変更してもファイルは置き換えられない。

selectshape の指定は省略可能。省略時には矩形とみなす。

realtime = true を指定して実行時にマッチング画像を取得するとき、同時に useredframe = true を指定したときにはパス指定マッチング画像と赤枠指定マッチング画像の両方を使用する。useredframe の指定は省略可能。省略した場合は false が指定されたとみなす。realtime = false かつ useredframe = true は指定可能だが、赤枠指定マッチング画像のみを使用する動作となる。

pathspecified の指定は省略可能。省略した場合はファイルパスが指定されたとみなす。

マウスアクションが状態チェックのとき、結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

マウスアクション mouseaction には次のいずれかを指定。

表 8-19 画像マッチングのマウスアクション

マウスアクション	説明
$ImageMatch.Check	状態チェック
$ImageMatch.LeftClick	左ボタンクリック
$ImageMatch.RightClick	右ボタンクリック
$ImageMatch.LeftDouble	左ボタンダブルクリック
$ImageMatch.RightDouble	右ボタンダブルクリック
$ImageMatch.Move	マウス移動のみ
$ImageMatch.LeftTriple	左ボタントリプルクリック
$ImageMatch.RightTriple	右ボタントリプルクリック
$ImageMatch.LeftClickDrag	左クリックし、マッチング箇所までドラッグ
$ImageMatch.RightClickDrag	右クリックし、マッチング箇所までドラッグ

縮尺率 scale には次のいずれかを指定。

表 8-20 画像マッチングの縮尺率

縮尺率	説明
$ImageMatch.Same	等倍
$ImageMatch.Half	1/2
$ImageMatch.Quarter	1/4

円形矩形モード selectshape には次のいずれかを指定。

表 8-21 円形矩形モード指定

円形矩形モード指定	説明
$ImageMatch.SelectShape_Rectangle	矩形モードを指定する
$ImageMatch.SelectShape_Ellipse	円形（楕円形）モードを指定する

開始点 startpoint には次のいずれかを指定。

表 8-22 画像マッチングの開始点

開始点	説明
$ImageMatch.StartPoint_LeftTop	左上
$ImageMatch.StartPoint_LeftBottom	左下
$ImageMatch.StartPoint_RightTop	右上
$ImageMatch.StartPoint_RightBottom	右下

座標指定 x_coordinate y_coordinate には次のいずれかを指定。

表 8-23 画像マッチングの座標指定

座標指定	説明
$ImageMatch.Coordinate_Direct	ピクセル単位で指定する
$ImageMatch.Coordinate_Percent	パーセントで指定する

パス指定マッチング画像指定 pathspecified には次のいずれかを指定。

表 8-24 パス指定マッチング画像指定

パス指定マッチング画像指定	説明
$ImageMatch.Path_File	ファイルパスを指定する
$ImageMatch.Path_Folder	フォルダパスを指定する

8.3.2 輪郭マッチング

```
WinActor.OutlineMatch [_OriginalID_f63cb690ce1d = 画像 ID 番号]
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    targetrange     = (x = X 位置, y = Y 位置, width = 幅, height = 高さ),  // 画像の探索部分
    mousecoordinate = (enable = true / false, x = X 位置, y = Y 位置),
                                                          // マウスアクション操作位置
    mouseaction     = マウスアクション,
    precision       = 精度,
    scale           = 縮尺率、
    timeout         = タイムアウト時間,     // ミリ秒
    searchrange      = (enable = true / false , x = X 位置, width = 幅, y = Y 位置, height =
高さ,
                        startpoint= 開始点, x_coordinate = 座標指定, y_coordinate = 座標指
定),
    realtime        = true / false,              // 実行時にマッチング画像を取得するとき true
    realtimefile     = ファイルパス文字列 または 変数名, // realtime = true の場合に、ファイ
```

```
ル名またはフォルダ名を指定する
    useredframe     = true / false,  // パス指定マッチング画像と赤枠指定マッチング画像の
                                     // 両方を使用するとき true
    pathspecified   = ファイル・フォルダ指定, // パス指定マッチング画像の指定方法
    imagesplit      = true / false,      // 分割マッチングの場合 true
    value           = 結果返却先変数名,
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

プリアンブルの _OriginalID_f63cb690ce1d 属性は WinActor で保持している画像を識別する ID を指定しているため、変更しないこと。

プリアンブルには、comment や name を追加可能。

画像ファイルは、WinActor のシナリオ編集画面で編集（変更、追加など）する必要がある。

WSS で名前を変更してもファイルは置き換えられない。

realtime = true を指定して実行時にマッチング画像を取得するとき、同時に useredframe = true を指定したときにはパス指定マッチング画像と赤枠指定マッチング画像の両方を使用する。useredframe の指定は省略可能。省略した場合は false が指定されたとみなす。realtime = false かつ useredframe = true は指定可能だが、赤枠指定マッチング画像のみを使用する動作となる。

pathspecified の指定は省略可能。省略した場合はファイルパスが指定されたとみなす。

マウスアクションが状態チェックのとき、結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

マウスアクション mouseaction には次のいずれかを指定。

表 8-25 輪郭マッチングのマウスアクション

マウスアクション	説明
$OutlineMatch.Check	状態チェック
$OutlineMatch.LeftClick	左ボタンクリック
$OutlineMatch.RightClick	右ボタンクリック
$OutlineMatch.LeftDouble	左ボタンダブルクリック
$OutlineMatch.RightDouble	右ボタンダブルクリック

マウスアクション	説明
$OutlineMatch.Move	マウス移動のみ
$OutlineMatch.LeftTriple	左ボタントリプルクリック
$OutlineMatch.RightTriple	右ボタントリプルクリック
$OutlineMatch.LeftClickDrag	左クリックし、マッチング箇所までドラッグ
$OutlineMatch.RightClickDrag	右クリックし、マッチング箇所までドラッグ

精度 precision には次のいずれかを指定。

表 8-26 輪郭マッチングの精度

精度	説明
$OutlineMatch.LowPrecision	低(速度優先)
$OutlineMatch.MiddlePrecision	中(標準)
$OutlineMatch.HighPrecision	高(精度優先)

縮尺率 scale には次のいずれかを指定。

表 8-27 輪郭マッチングの縮尺率

縮尺率	説明
$OutlineMatch.Same	等倍
$OutlineMatch.Half	1/2
$OutlineMatch.Quarter	1/4

開始点 startpoint には次のいずれかを指定。

表 8-28 輪郭マッチングの開始点

開始点	説明
$OutlineMatch.StartPoint_LeftTop	左上
$OutlineMatch.StartPoint_LeftBottom	左下
$OutlineMatch.StartPoint_RightTop	右上
$OutlineMatch.StartPoint_RightBottom	右下

座標指定 x_coordinate y_coordinate には次のいずれかを指定

表 8-29 輪郭マッチングの座標指定

座標指定	説明
$OutlineMatch.Coordinate_Direct	ピクセル単位で指定する
$OutlineMatch.Coordinate_Percent	パーセントで指定する

パス指定マッチング画像指定 pathspecified には次のいずれかを指定。

表 8-30 パス指定マッチング画像指定

パス指定マッチング画像指定	説明
$OutlineMatch.Path_File	ファイルパスを指定する
$OutlineMatch.Path_Folder	フォルダパスを指定する

8.3.3 OCR マッチング

```
WinActor.OCRMatch [_OriginalID_f63cb690ce1d = 画像 ID 番号]
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    targetrange     = (x = X 位置, y = Y 位置, width = 幅, height = 高さ),  // 画像の探索部分
    mousecoordinate = (enable = true / false, x = X 位置, y = Y 位置),
                                                          // マウスアクション操作位置
    mouseaction     = マウスアクション,
    timeout         = タイムアウト時間,     // ミリ秒
    searchrange      = (enable = true / false , x = X 位置, width = 幅, y = Y 位置, height =
高さ,
                        startpoint= 開始点, x_coordinate = 座標指定, y_coordinate = 座標指
定),
    ocrmatchingtext = マッチング文字列返却変数名,
    value           = 結果返却先変数名,
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

プリアンブルの _OriginalID_f63cb690ce1d 属性は WinActor で保持している画像を識別する ID を指定しているため、変更しないこと。

プリアンブルには、comment や name を追加可能。

画像ファイルは、WinActor のシナリオ編集画面で編集（変更、追加など）する必要がある。

WSS で名前を変更してもファイルは置き換えられない。

マウスアクションが状態チェックのとき、結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

マウスアクション mouseaction には次のいずれかを指定。

表 8-31 OCR マッチングのマウスアクション

マウスアクション	説明
$OCRMatch.Check	状態チェック
$OCRMatch.LeftClick	左ボタンクリック
$OCRMatch.RightClick	右ボタンクリック
$OCRMatch.LeftDouble	左ボタンダブルクリック
$OCRMatch.RightDouble	右ボタンダブルクリック
$OCRMatch.Move	マウス移動のみ
$OCRMatch.LeftTriple	左ボタントリプルクリック
$OCRMatch.RightTriple	右ボタントリプルクリック
$OCRMatch.LeftClickDrag	左クリックし、マッチング箇所までドラッグ
$OCRMatch.RightClickDrag	右クリックし、マッチング箇所までドラッグ

開始点 startpoint には次のいずれかを指定。

表 8-32 OCR マッチングの開始点

開始点	説明
$OCRMatch.StartPoint_LeftTop	左上
$OCRMatch.StartPoint_LeftBottom	左下
$OCRMatch.StartPoint_RightTop	右上
$OCRMatch.StartPoint_RightBottom	右下

座標指定 x_coordinate y_coordinate には次のいずれかを指定。

表 8-33 OCR マッチングの座標指定

座標指定	説明
$OCRMatch.Coordinate_Direct	ピクセル単位で指定する
$OCRMatch.Coordinate_Percent	パーセントで指定する

8.3.4　ウィンドウ状態待機

```
WinActor.WindowStateWait プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    win_state       = 画面変化,
    state           = 待機種別,
    timeout         = タイムアウト時間,
                              // ミリ秒　「待機種別」で一定時間待つことを指定した場合に有効
    value           = 結果返却先変数名,
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置
Y)
  )
```

画面変化 win_state には次のいずれかを指定。

表 8-34 ウィンドウ状態待機の画面変化

画面変化	説明
$Window.Front	画面が手前になるまで
$Window.Behind	画面が手前でなくなるまで
$Window.Enable	画面が操作可能になるまで
$Window.Disable	画面が操作不可能になるまで
$Window.Appear	画面が表示されるまで
$Window.Disappear	画面が消えるまで

待機種別 state には次のいずれかを指定。

表 8-35 ウィンドウ状態待機の待機種別

待機種別	説明
$Window.WaitFor	一定時間待つ
$Window.CheckOnly	状態取得のみ

true または false の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.5 指定時間待機

```
WinActor.TimerWait プリアンブル
  (
    mode         = モード,
    timeout      = 待ち時間,          // ミリ秒数値または変数名。モードが「指定時間待つ」の場合
    wait_time    = 待機時刻,          // 文字列または変数名。モードが「指定時刻まで待つ」の場合
    check_time   = 状態比較時刻,      // 文字列または変数名。モードが「指定時刻との比較」の場合
    check_value = 比較結果返却変数名,
    date_format = 日付形式,
    timezone     = タイムゾーン
  )
```

モード mode には次のいずれかを指定。

表 8-36 指定時間待機のモード

モード	説明
$TimerWait.Sleep	指定時間待つ
$TimerWait.Until	指定時刻まで待つ
$TimerWait.Check	指定時刻との比較

日付形式 date_format には次のいずれか、または WinActor で許されている日付形式文字列を指定。

表 8-37 指定時間待機の日付形式

日付形式	説明
$TimerWait.ScenarioInfoDateFormat	シナリオ情報
$TimerWait.OptionInfoDateFormat	オプション画面

タイムゾーン timezone には次のいずれか、または WinActor で許されているタイムゾーン文字列を指定。

表 8-38 指定時間待機のタイムゾーン

タイムゾーン	説明
$TimerWait.ScenarioInfoTimeZone	シナリオ情報
$TimerWait.OptionInfoTimeZone	オプション画面
$TimerWait.DefaultTimeZone	OS デフォルト

mode が指定時刻との比較の場合、true または false の結果を check_value で指定した変数に返す。

詳しくは『WinActor 操作マニュアル』の『指定時間待機』ノードの『時刻指定フォーマット』を参照のこと。

代入や式の中に現れた場合は、比較結果返却変数名には結果は格納しない。比較結果返却変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには比較結果返却変数名の指定は省略される。

8.3.6 文字列送信

```
WinActor.SendText プリアンブル
  (
    window_rule_ref = Window_rule ウィンドウ識別名,
    control         = (instance<チェック> = 式, text<チェック> = 式, position<チェック> = ポジション),
    value           = 文字列 または 変数名,
    sendcr          = true または false,
    verify          = true または false,
    capture         = (imageid = イメージ ID 文字列, x = マウス操作位置 X, y = マウス操作位置 Y)
  )
```

値は返さない。

value には、送信する文字列、または送信内容を格納した変数名を指定する。指定しないときには無名識別子 '' を置く。

value 属性の定数は、数値でも文字列に変換する。

sendcr と verify には true または false を指定する。指定を省略したときには false を指定したとする。

true または false 以外を指定したときには警告となる。false を指定したとみなして続行する。

sendcr と verify の意味は次のとおり。

表 8-39 sendcr と verify

アクション列	説明
sendcr	リターンキーを送信
verify	送信結果を検証（検証エラー時は一時停止）

8.3.7　コマンド実行

```
WinActor.Launcher プリアンブル
  (
    command      = コマンド名 または コマンド名を格納した変数名,
    option       = オプション文字列 または オプション文字列を格納した変数名,
    execute_mode = 実行モード,
    set_value    = 結果返却先変数名
  )
```

実行モード execute_mode には次のいずれかを指定する。

表 8-40 コマンド実行の実行モード

実行モード	説明
$Launcher.Single	起動のみ。追加起動しない。
$Launcher.Multi	起動のみ。追加起動する。
$Launcher.WaitForEnd	終了まで待つ。結果格納。

execute_mode＝$Launcher.WaitForEnd の場合、値を set_value で指定した変数に返す。代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.8　スクリプト実行

```
WinActor.Script プリアンブル
  (
     window_rule_ref       = Window_rule ウィンドウ識別名,
     environment_type      = 変数共有指定,
     library_provider      = ライブラリの提供者（文字列),
     library_id            = ライブラリの ID（文字列),
     library_version       = ライブラリバージョン（文字列),
     library_original_name = ライブラリ名（文字列),
     edit_lock             = true または false,

     スクリプトパラメータ記述 = 記述によって 値 または 変数名,
        // パラメータの個数だけこの行を並べる

     /*** Note ***/
     note = @"""
注釈記述
""",
```

```
    /*** Note End ***/
    /*** Script ***/
    script = @"""
スクリプト記述
"""
    /*** Script End ***/
 )
```

注釈とスクリプトは @""" で終わる行から、""" で始まる行までの間に記述すること。

スクリプトパラメータ記述は '' （シングルクオート）で囲って識別子とすること。
「パラメータ名」はスクリプト記述内の WinActor との値のやり取りのための名前となる。

表 8-41 スクリプト実行のスクリプトパラメータ記述

スクリプトパラメータ記述	説明
'!パラメータ名!'	変数名もしくは値
'!パラメータ名|項目 1,項目 2,項目 3...!'	プルダウンメニュー
'$パラメータ名$'	変数名
'@パラメータ名@'	ウィンドウ識別名
'!パラメータ名|FILE|!'	ファイル選択ダイアログ
'!パラメータ名|FILE:EXCEL|!'	ファイル選択ダイアログ(Excel)
'!パラメータ名|FILE:ZIP|!'	ファイル選択ダイアログ(zip)
'!パラメータ名|FILE:CSV|!'	ファイル選択ダイアログ(csv)
'!パラメータ名|FILE:IMG|!'	ファイル選択ダイアログ(画像)

変数共有指定 environment_type には次のいずれかを指定する。

表 8-42 スクリプト実行の変数共有指定

変数共有指定	説明
$Script.EnvIndependent	スクリプト実行ごとに変数を設定する
$Script.EnvShared	スクリプト実行間で変数を共有する

各サンプルライブラリのスクリプト実行ノードは、バージョン情報パラメータ(library_provider, library_id, library_version, library_original_name)を持つ。これらのパラメータについては、WSS では変更できない。

edit_lock の指定を省略したときには false を指定したものとみなす。

WinActor 本体で編集ロック(edit_lock)されたスクリプト実行ノードについては、WSS ではスクリプト記述 が表示されない。変更することもできない。

値は返さない。

例

```
WinActor.Script [name = "カウントダウン", comment = ""]
(
    window_rule_ref = "",
    environment_type = $Script.EnvIndependent,
    library_provider = "NTT アドバンステクノロジ株式会社",
    library_id = "AT05001L",
    library_version = "1.0.0",
    library_original_name = "カウントダウン",
    edit_lock = false,
    '$カウンタ$' = a,
    /*** Note ***/
    note = @"""
指定した変数の数値をカウントダウンします。

カウンタ：カウントダウン対象の変数名を指定してください。
""",
    /*** Note End ***/
    /*** Script ***/
    script = @"""
c = GetUMSVariable( $カウンタ$ )
ci = int(c)
ci = ci - 1
SetUMSVariable $カウンタ$, ci
"""
    /*** Script End ***/
);
```

8.3.9 Python 実行

```
WinActor.PythonScript プリアンブル
  (
    window_rule_ref       = Window_rule ウィンドウ識別名,
    environment_type      = 変数共有指定,
    library_provider      = ライブラリの提供者（文字列),
    library_id            = ライブラリの ID（文字列),
    library_version       = ライブラリバージョン（文字列),
    library_original_name = ライブラリ名（文字列),
    edit_lock             = true または false,
```

```
    スクリプトパラメータ記述 = 記述によって 値 または 変数名,
       // パラメータの個数だけこの行を並べる

    /*** Note ***/
    note = @"""
注釈記述
""",
    /*** Note End ***/
    /*** Script ***/
    script = @""""
Python スクリプト記述
""""
    /*** Script End ***/
  )
```

注釈は @"""（ダブルクオート 3 個）で終わる行から、"""（ダブルクオート 3 個）で始まる行までの間に記述すること。

Python スクリプトは @""""""（ダブルクオート 4 個）で終わる行から、""""""（ダブルクオート 4 個）の行までの間に記述して python 文字列との衝突を避ける。

スクリプトパラメータ記述は ''（シングルクオート）で囲って識別子とすること。

「パラメータ名」は Python スクリプト記述内の WinActor との値のやり取りのための名前となる。『表 8-41 スクリプト実行のスクリプトパラメータ記述』と同一。

表 8-43 Python 実行のスクリプトパラメータ記述

スクリプトパラメータ記述	説明
'!パラメータ名!'	変数名もしくは値
'!パラメータ名|項目 1,項目 2,項目 3...!'	プルダウンメニュー
'$パラメータ名$'	変数名
'@パラメータ名@'	ウィンドウ識別名
'!パラメータ名|FILE|!'	ファイル選択ダイアログ
'!パラメータ名|FILE:EXCEL|!'	ファイル選択ダイアログ(Excel)
'!パラメータ名|FILE:ZIP|!'	ファイル選択ダイアログ(zip)
'!パラメータ名|FILE:CSV|!'	ファイル選択ダイアログ(csv)
'!パラメータ名|FILE:IMG|!'	ファイル選択ダイアログ(画像)

変数共有指定 environment_type には次のいずれかを指定する。

表 8-44 Python 実行の変数共有指定

変数共有指定	説明
$Script.EnvIndependent	Python スクリプト実行ごとに変数を設定する
$Script.EnvShared	Python スクリプト実行間で変数を共有する

サンプルライブラリの Python 実行ノードは、バージョン情報パラメータ(library_provider, library_id, library_version, library_original_name)を持つ。これらのパラメータについては、WSS では変更できない。

edit_lock の指定を省略したときには false を指定したものとみなす。

WinActor 本体で編集ロック(edit_lock)された Python 実行ノードについては、WSS では スクリプト記述 が表示されない。変更することもできない。

値は返さない。

例

```
WinActor.PythonScript [name = "文字列連結", comment = ""]
(
    window_rule_ref = "",
    environment_type = $Script.EnvIndependent,
    library_provider = "",
    library_id = "",
    library_version = "",
    library_original_name = "",
    edit_lock = false,
    '!文字列 1!' = "first",
    '!文字列 2!' = "second",
    '$連結結果$' = result_py,      /*** Note ***/
    note = @"""
2 つの文字列を連結します。
""",
    /*** Note End ***/
    /*** Script ***/
    script = @"""""
result = "".join([!文字列 1!,  !文字列 2!])
vm_result = $連結結果$

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

8.3.10 Excel 操作

```
WinActor.Excel プリアンブル
  (
    operation    = エクセル操作,
    file_path    = エクセルファイルのパスまたは変数名,  // いずれの操作でも必要
    sheet        = シート名または変数名,              // 操作が値の設定または取得の場合必要
    cell         = セル指定または変数名,              // 操作が値の設定または取得の場合必要
    source_value = 設定する値または変数名,             // 操作が値の設定の場合必要
    target_value = 結果返却先変数名,                  // 操作が値の取得の場合必要
    macro        = マクロ名または変数名                // 操作がマクロ実行の場合必要
  )
```

エクセル操作 operation には次のいずれかを指定する。

表 8-45 Excel 実行のエクセル操作

エクセル操作	説明
$Excel.GetValue	値の取得
$Excel.SetValue	値の設定
$Excel.RunMacro	マクロ実行

「または変数名」は、それぞれ必要な情報を格納した変数の名前を意味する。

操作が値の取得の場合、取得した値を target_variable で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.11 クリップボード

```
WinActor.Clipboard プリアンブル
  (
     mode = 動作,
     set_value = 式,       // 動作が $ClipBoard.Set の場合に必要
     get_value = 結果返却先変数名
  )
```

動作 mode には次のいずれかを指定する

表 8-46 クリップボードの動作

実行モード	説明
$ClipBoard.Set	値をクリップボードに設定する。
$ClipBoard.Get	クリップボードの内容を取得する。

クリップボードへの設定の場合、get_value は省略可能。

クリップボード内容取得の場合、内容を get_value で指定した変数に返す。set_value は省略可能。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

次の WinActor.SetToClipboard と WinActor.GetFromClipboard を使用すると引数が少なくてわかりやすい。

8.3.12 クリップボードに設定

```
WinActor.SetToClipboard プリアンブル
  (
      文字列 または クリップボードに設定する内容を格納した変数名
  )
```

値は返さない。

8.3.13 クリップボードの値を取得

```
WinActor.GetFromClipboard プリアンブル
  (
    get_value = 結果返却先変数名
  )
```

値を get_value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能だが () は必要。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

例

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

8.3.14 待機ボックス

```
WinActor.WaitBox プリアンブル
  (
    mode    = モード,
    message = メッセージ文字列 またはメッセージを格納した変数名
  )
```

モード mode には次のいずれかを指定する。

表 8-47 待機ボックスのモード

モード	説明
$WaitBox.Confirm	確認待ち(OK ボタンのみを表示)
$WaitBox.Query	問い合わせ(継続、停止ボタンを表示)

値は返さない。

8.3.15 インプットボックス

```
WinActor.InputBox プリアンブル
  (
    message = メッセージ文字列 または メッセージを格納した変数名,
    value = 結果返却先変数名
  )
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

message 属性の定数は、数値でも文字列として扱う。

8.3.16 選択ボックス

```
WinActor.SelectBox プリアンブル
  (
    message = メッセージ文字列 または メッセージを格納した変数名,
    items = 選択候補,
    value = 結果返却先変数名
  )
```

選択候補 items は、候補文字列を丸括弧で囲んで指定する。

文字列、または文字列を格納した定数が使用できる。

数値は文字列として扱う。

例:

```
items = ("赤", "青", "白"),
items = (1, 2, 3),
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.17 音（ブザー）

```
WinActor.Beep プリアンブル ()
```

ブザー音を発する。

引数はない。

値は返さない。

8.3.18 音（WAVE ファイル）

```
WinActor.Speaker  [name = "名前", comment = "コメント", _OriginalID_f63cb690ce1d = 268]
  (
    selectFile = WAVE ファイル名,
    wait = true / false       // 再生が終わるまで待機するとき true
  )
```

プリアンブルの _OriginalID_f63cb690ce1d 属性は WinActor で保持している音ファイルを識別する ID を指定しているため、変更しないこと。

プリアンブルには、comment や name を追加可能。

WAVE ファイルは、WinActor のシナリオ編集画面で編集（変更、追加など）する必要がある。

WSS で名前を変更してもファイルは置き換えられない。

値は返さない。

8.3.19 変数値設定 SetVariable

```
結果返却先変数名 = 式 プリアンブル ;
```

変数値の設定は、代入文を使って記述する。

プリアンブルで、name 名前、comment コメントを設定できる。

アクションとして記述する場合は、次のとおり。

```
WinActor.SetVariable プリアンブル（
  val   = 定数式,
  value = 結果返却先変数名
）
```

定数式の結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

代入で記述したときには、「val = 」も省略可能

例: 次のいずれも同じ結果となる。

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

8.3.20 変数値コピー CopyVariable

```
結果返却先変数名 = 式 プリアンブル ;
```

変数値のコピーは、代入文を使って記述する。

プリアンブルで、name 名前、comment コメントを設定できる。

アクションとして記述する場合は、次のとおり。

```
WinActor.CopyVariable プリアンブル
（
   from = コピー元変数名,
   to   = コピー先変数名,
）
```

コピー元変数の内容を、to で指定した変数に設定する。

代入や式の中に現れた場合は、to で与えた変数には結果は格納しない。to の指定は省略可能。

代入で記述したときには、「from = 」も省略可能

例: 次のいずれも同じ結果となる。

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

8.3.21 日時取得

```
WinActor.GetDateTime プリアンブル
  (
    format      = フォーマットタイプ,
    date_format = 日付形式,
    timezone    = タイムゾーン,
    value       = 結果返却先変数名
  )
```

フォーマットタイプ format には以下のいずれかを指定。

表 8-48 日時取得のフォーマットタイプ

フォーマットタイプ	説明
$GetDateTime.DateTime	日付と時間
$GetDateTime.Date	日付のみ
$GetDateTime.Time	時間のみ

日付形式 date_format には次のいずれか、または WinActor で許されている日付形式文字列を指定。

表 8-49 日時取得の日付形式

日付形式	説明
$GetDateTime.ScenarioInfoDateFormat	シナリオ情報（省略時の設定）
$GetDateTime.OptionInfoDateFormat	オプション画面

タイムゾーン timezone には次のいずれか、または WinActor で許されているタイムゾーン文字列を指定。

表 8-50 日時取得のタイムゾーン

タイムゾーン	説明
$GetDateTime.ScenarioInfoTimeZone	シナリオ情報（省略時の設定）

タイムゾーン	説明
$GetDateTime.OptionInfoTimeZone	オプション画面
$GetDateTime.DefaultTimeZone	OS デフォルト

date_format と timezone は省略可能。

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.22 ユーザ名取得

```
WinActor.GetUserName プリアンブル
  (
    value = 結果返却先変数名
  )
```

結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能だが () は必要。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

例

```
uname = WinActor.GetUserName [Comment = "ユーザ名取得"] ();
```

8.3.23 四則演算

四則演算は式と代入文を使って記述する。

アクションとして記述する場合は、次のとおり。

```
WinActor.Calculate プリアンブル (
  operator = 演算,
  left  = 二項演算の引数,
  right = 二項演算の引数,
  value = 結果返却先変数名
)
```

演算 operator には、次のいずれかを指定する。

表 8-51 四則演算の演算

演算
$Calculate.Plus
$Calculate.Minus
$Calculate.Mul
$Calculate.Div

二項演算の引数には数値、変数名、式を記述することができる。

式を記述したときにはコンパイラが式の演算用のノードを自動生成する。

演算結果を value で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.3.24 カウントアップ

カウントアップは式と代入文を使って記述する。

アクションとして記述する場合は、次のとおり。

```
WinActor.CountUP プリアンブル（
  value = 変数名,
  add = 増加分定数式
）
```

値は返さない。

value で指定した変数に定数を加える。

定数に負の値を指定すると WinActor 本体で警告が出る。

8.3.25 全角化/半角化

```
WinActor.TextConvert プリアンブル
  （
    operation = 操作,
    target_variable = 変換対象文字列保持変数名
  ）
```

操作 operation には次のいずれかを指定する。

表 8-52 全角化/半角化の操作

操作	説明
$FullHalfWidth.ToFullWidth	全角文字に統一する
$FullHalfWidth.ToHalfWidth	半角文字に統一する

値は返さない。

8.3.26 イベント監視

```
WinActor.EventsWatch プリアンブル ()
```

Events に設定されたイベントトリガーを監視し、検知した場合に呼び出し処理を実行する。
引数はない。

8.3.27 イベント監視登録

```
WinActor.EventAdd プリアンブル ( Events イベント監視名 )
```

Events に設定されたイベントを、イベント監視の対象として登録する。
登録するイベント監視名を引数に指定する。

8.3.28 イベント監視解除

```
WinActor.EventRemove プリアンブル ( Events イベント監視名 )
```

Events に設定されたイベントを、イベント監視の対象から解除する。
解除するイベント監視名を引数に指定する。

8.3.29 イベント監視終了

```
WinActor.EventsIgnore プリアンブル ()
```

イベント監視を終了する。
引数はない。

8.4　WinActor メール管理、HTTP 関連、JSON

8.4.1　メール受信設定

```
WinActor.MailReceiveSet プリアンブル
    (
        mail_rule_info = メール取得条件,
        host_name = 文字列 または 変数,      // メール受信サーバホスト名
        user = 文字列 または 変数,           // メール受信サーバに接続するユーザ名
        pass = 文字列 または 変数,           // メール受信サーバに接続するパスワード
        auth_type   = 認証タイプ,            // 下記参照
        port        = ポート番号,            // 整数値 110 など
        conn_time   = 接続タイムアウト,      // ミリ秒 10000 など（10 秒）
        cmd_time    = 受信タイムアウト,      // ミリ秒 10000 など（10 秒）
        mail_input  = 文字列 または 変数,    // メール保存場所
        is_del_mail = true / false,          // 受信時、サーバからメールを消去するか
        attach_save = true / false,          // 添付ファイルを保存するか
        extention_not_save = true / false,
                                        // 空白区切り extention_not_save が true のとき有効
        except_extension = "*.exe *.bat *.vbs *.msi *.jar",
                                        // 保存しない添付ファイル拡張子を指定
        security_type = 接続保護タイプ       // 下記参照
    );
```

メール受信設定を行う。

値は返さない。

認証タイプ auth_type には次のいずれかを指定する

表 8-53 メール受信設定の認証タイプ

認証タイプ	説明
$MailAuth.UserPass	USER/PASS 認証
$MailAuth.APOP	APOP 認証

接続保護タイプ security_type には次のいずれかを指定する

表 8-54 メール受信設定の接続保護タイプ

接続保護タイプ	説明
$MailSecurity.No	保護なし
$MailSecurity.TLS_SSL	保護あり POP3S
$MailSecurity.StartTLS	保護あり StartTLS

■　メール取得条件 mail_rule_info

メールを取得する条件を記述する。条件はカンマで区切って複数記述できる。すべての条件を満たしたメールのみを取得する。

条件の構文は次のとおり。

```
(item = 項目, cond = 条件, value = 値)
または
(item = 項目, cond = 条件, var = 変数名)
```

項目 item には次のいずれかを指定する

表 8-55 メール取得条件の項目

項目	説明
$MailRule.To	宛先
$MailRule.From	差出人
$MailRule.Subject	件名

条件 cond には次のいずれかを指定する

表 8-56 メール取得条件の条件

条件	説明
$MailRule.Include	含む
$MailRule.AtFirst	先頭に含む
$MailRule.AtLast	最後に含む
$MailRule.Equal	完全一致
$MailRule.Regex	正規表現

例　差出人に example.com が含まれ、かつ、件名に report が含まれているメールを受信する。

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

8.4.2　メール受信

```
WinActor.MailReceive プリアンブル
```

```
    (
        get_method = 取得方法,
        no_receiver_mail = 受信メールなしの場合の動作,
        get_mail_num = 結果返却先変数名
    )
```

メールを受信する。

取得方法 get_method には次のいずれかを指定する

表 8-57 メール受信の取得方法

取得方法	説明
$MailReceive.OneByOne	1 件ずつ取得
$MailReceive.GetAll	全取得
$MailReceive.NumOnly	メール数取得のみ

受信メールなしの場合の動作 no_receiver_mail には次のいずれかを指定する

表 8-58 メール受信の受信メールなしの場合の動作

受信メールなしの場合の動作	説明
$MailReceive.Wait	受信待ち受信できるまで待機
$MailReceive.Error	エラー応答を返却
$MailReceive.MailNum	メール数 0 を返す

取得メール数を get_mail_num で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.4.3 メール選択

```
WinActor.MailSelect プリアンブル
    (
        select_type = 選択種別,
        line_num = 行番号格納変数名,
        not_mail_error_return = true または false
    )
```

受信したメールを選択状態にする。

選択種別 select_type には次のいずれかを指定する

表 8-59 メール選択の選択種別

選択種別	説明
$MailSelect.Top	先頭行を選択
$MailSelect.NoProcessedTop	未処理の先頭行を選択
$MailSelect.ProcessedTop	処理済みの先頭行を選択
$MailSelect.Next	次の行を選択
$MailSelect.NextNoProcessed	次の未処理を選択
$MailSelect.NextProcessed	次の処理済みの行を選択

選択した行番号を line_num で指定した変数に返す。

代入や式の中に現れた場合は、 行番号格納変数名には結果は格納しない。行番号格納変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには行番号格納変数名の指定は省略される。

8.4.4 選択中メールの状態変更

```
WinActor.MailStatusChg プリアンブル
  (
    $MailStatusChg.Processed または $MailStatusChg.NoProcessed
  )
```

引数に $MailStatusChg.Processed または $MailStatusChg.NoProcessed を指定する。

それぞれ処理済み、未処理、に設定することを意味する。

値は返さない。

8.4.5 メールフォルダ同期

```
WinActor.MailSync プリアンブル ()
```

メール管理画面と実際に受信したメール情報とを同期させる。

『WinActor メール受信シナリオ作成マニュアル』を参照のこと。

引数はない。

値は返さない。

8.4.6　メール処理済み削除

```
WinActor.MailRemoveProcessed プリアンブル
 (
    deleted_mail_num = 結果返却先変数名
 )
```

削除したメール件数を deleted_mail_num で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.4.7　メール削除

```
WinActor.MailRemove プリアンブル ()
```

選択中メールを削除する。

引数はない。

値は返さない。

8.4.8　メール情報コピー

```
WinActor.MailCopyClip プリアンブル
  ( 引数 )
```

選択中メールの情報をクリップボードにコピーする。

値は返さない。

引数に以下のいずれかを指定する。

表 8-60 メール情報コピーの引数

引数	説明
$MailCopyClip.UniqueID	ユニーク ID
$MailCopyClip.FolderName	フォルダ名
$MailCopyClip.Status	状態(未処理/処理済)
$MailCopyClip.SendDate	送信日
$MailCopyClip.From	差出人
$MailCopyClip.Subject	件名
$MailCopyClip.Body	本文

引数	説明
$MailCopyClip.NumberOfAttached	添付ファイル数

8.4.9 添付ファイル名取得

```
WinActor.MailAttachName プリアンブル
    (
        attach_file_number = 添付ファイル番号,
        attach_file_name = 名前格納変数名
    );
```

添付ファイル名を attach_file_name で指定した変数に返す。

代入や式の中に現れた場合は、名前格納変数名には結果は格納しない。名前格納変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには名前格納変数名の指定は省略される。

8.4.10 メール情報取得

```
WinActor.MailGetInfo プリアンブル
    (
        uid        = ユニーク ID 格納変数名,
        dir        = フォルダ名格納変数名,
        stat       = 状態（未処理/処理済）格納変数名,
        send_date  = 送信日格納変数名,
        from       = 差出人格納変数名,
        to         = 受取人格納変数名,
        cc         = カーボンコピー受取人格納変数名,
        subject    = 件名格納変数名,
        message    = 本文格納変数名,
        attachment = 添付ファイル数格納変数名
    );
```

受信したメールの情報を取得する。

情報が不要な項目には変数名として無名識別子 '' を指定するか、項目を省略する。

情報は指定した変数に返し、全体としての値は返さない。

8.4.11 メール受信設定インポート

```
WinActor.MailReceiveImport プリアンブル
  (
```

```
    ファイルパスまたはファイルパスを格納した変数名
)
```

メール受信設定内容をインポートする。

値は返さない。

8.4.12 Gmail 受信設定

```
WinActor.GmailReceiveSet プリアンブル
    (
        mail_rule_info = メール取得条件,
        conn_time    = 接続タイムアウト,      // ミリ秒 10000 など（10 秒）
        cmd_time     = 受信タイムアウト,      // ミリ秒 10000 など（10 秒）
        mail_input   = 文字列 または 変数,    // メール保存場所
        attach_save = true / false,          // 添付ファイルを保存するか
        extention_not_save = true / false,
                                        // 空白区切り extention_not_save が true のとき有効
        except_extension = "*.exe *.bat *.vbs *.msi *.jar",
                                        // 保存しない添付ファイル拡張子を指定
    );
```

Gmail の受信設定を行う。

接続タイムアウトと受信タイムアウトの最小値は 100、最大値は 3,600,000 ミリ秒である。

値は返さない。

■　メール取得条件 mail_rule_info

メールを取得する条件を記述する。条件はカンマで区切って複数記述できる。すべての条件を満たしたメールのみを取得する。

条件の構文は次のとおり。

```
(item = 項目, cond = 条件, value = 値)
または
(item = 項目, cond = 条件, var = 変数名)
```

項目 item には次のいずれかを指定する

表 8-61 メール取得条件の項目

項目	説明
$GmailRule.To	宛先
$GmailRule.From	差出人

項目	説明
$GmailRule.Subject	件名

条件 cond には次のいずれかを指定する

表 8-62 メール取得条件の条件

条件	説明
$GmailRule.Include	含む
$GmailRule.AtFirst	先頭に含む
$GmailRule.AtLast	最後に含む
$GmailRule.Equal	完全一致
$GmailRule.Regex	正規表現

例　差出人に example.com が含まれ、かつ、件名に report が含まれているメールを受信する。

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

8.4.13 Gmail 受信

```
WinActor.GmailReceive プリアンブル
    (
        get_method = 取得方法,
        no_receiver_mail = 受信メールなしの場合の動作,
        get_mail_num = 結果返却先変数名
    )
```

Gmail を受信する。

取得方法 get_method には次のいずれかを指定する

表 8-63 Gmail 受信の取得方法

取得方法	説明
$GmailReceive.OneByOne	1 件ずつ取得
$GmailReceive.GetAll	全取得
$GmailReceive.NumOnly	メール数取得のみ

受信メールなしの場合の動作 no_receiver_mail には次のいずれかを指定する

表 8-64 Gmail 受信の受信メールなしの場合の動作

受信メールなしの場合の動作	説明
$GmailReceive.Wait	受信待ち受信できるまで待機
$GmailReceive.Error	エラー応答を返却
$GmailReceive.MailNum	メール数 0 を返す

取得メール数を get_mail_num で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

8.4.14 Gmail 送信設定

```
WinActor.GmailSendSet プリアンブル
    (
        conn_time    = 接続タイムアウト,      // ミリ秒 10000 など（10 秒）
        cmd_time     = 送信タイムアウト       // ミリ秒 10000 など（10 秒）
    );
```

Gmail の送信設定を行う。

接続タイムアウトと送信タイムアウトの最小値は 100、最大値は 3,600,000 ミリ秒である。

値は返さない。

8.4.15 Gmail 送信

```
WinActor.GmailSend プリアンブル
    (
       recipient_name = 文字列 または 変数名,  // 宛名
       recipient_address = 文字列 または 変数名,  //宛先メールアドレス
       subject = 文字列 または 変数名,  //件名
       body = 文字列 または 変数名  //本文
    );
```

Gmail を送信する。

値は返さない。

8.4.16 HTTP

```
WinActor.Http プリアンブル
  (
     method         = メソッド,
     url            = 接続先 URL 文字列 または 変数名,
     server_timeout = 接続タイムアウト,          // ミリ秒数値
     res_timeout    = レスポンスタイムアウト,  // ミリ秒数値

     req_header     = (キー = 値または変数名, ...),
     req_body_file  = 要求内容格納ファイルパス または 変数名,
     req_body       = (キー<型名> = 値または変数名, ...),
     req_use_file   = true / false,  // true のとき req_body_file を採用する

     res_header     = (キー = 格納先変数名, ...),
     res_body_file  = 格納先ファイルパス文字列 または 変数名,
     res_body       = (キー = 格納先変数名, ...)
     res_use_file   = true / false,  // true のとき res_body_file を採用する
     status_code    = 結果返却先変数名
  )
```

メソッド method には次のいずれかを指定する

表 8-65 HTTP のメソッド

メソッド
$HTTP.Get
$HTTP.Put
$HTTP.Post
$HTTP.Delete
$HTTP.Patch

req_body_file と req_body はどちらか 1 つを指定する。両方指定した場合は req_use_file が true の場合 req_body_file を false のとき req_body を使用する。両方指定して req_use_file を省略したときには req_body_file の指定を採用する。

res_body_file と res_body はどちらか 1 つを指定する。両方指定した場合は res_use_file が true の場合 res_body_file を false のとき res_body を使用する。両方指定して res_use_file を省略したときには res_body_file の指定を採用する。

メソッドに$HTTP.Get を指定した場合は、req_body_file と req_body は無視される。

メソッド毎に有効な引数（〇）と無視される引数（×）は次のとおり。

表 8-66 HTTP のメソッドと引数

引数	メソッド					説明
	Get	Put	Post	Delete	Patch	
req_header	○	○	○	○	○	キー ＝ 値または変数名
req_body_file	×	○	○	○	○	要求内容格納ファイルパス または 変数名
req_body	×	○	○	○	○	キー<型名> ＝ 値または変数名。JSON オブジェクトに変換して要求する。
res_header	○	○	○	○	○	キー ＝ 格納先変数名
res_body_file	○	○	○	○	○	格納先ファイルパス文字列 または 変数名
res_body	○	○	○	○	○	キー ＝ 格納先変数名。結果が JSON 形式であること。

型名には次のいずれかを指定する

表 8-67 HTTP の型名

型名	説明
integer	整数値
float	小数値
string	文字列
object	オブジェクト
array	配列
boolean	真偽値
null	Null 値

値を status_code で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

詳細や注意事項については、『WinActor ユーザライブラリサンプル説明書』の『HTTP』を参照のこと。

8.4.17 HTTP（詳細）

```
WinActor.Http2 プリアンブル
  (
    method       = メソッド,
```

```
    url          = 接続先 URL 文字列 または 変数名,
    proxy_mode   = true / false,  // オプション設定のプロキシサーバを使用するか
    response_timeout = レスポンスタイムアウト,  // ミリ秒数値
    auth         = true / false,  // BASIC 認証を使用するか
    auth_user    = ユーザ名文字列 または 変数名,
    auth_pass    = パスワード文字列 または 変数名,
    req_header   = 要求ヘッダ JSON 文字列 または 変数名,
    req_header_file = 要求ヘッダ内容格納ファイルパス または 変数名,
    req_param_multipart = true / false,   //  パラメータを multipart で送信するか
    req_param    = パラメータ JSON 文字列 または 変数名,
    req_param_file = パラメータ内容格納ファイルパス または 変数名,
    req_body     = 要求ボディ文字列,
    req_body_file = 要求ボディ内容格納ファイルパス,
    req_cookie   = 要求ヘッダクッキーJSON 文字列 または 変数名,
    req_cookie_file = 要求ヘッダクッキー内容格納ファイルパス または 変数名,
    req_fileupload_file = ファイルアップロード設定用ファイルパス または 変数名,
    fileupload_info = アップロードファイルリスト,
    res_header   = 応答ヘッダ格納先変数名,
    res_header_file = 応答ヘッダ格納先ファイルパス または 変数名,
    res_header_filetype = 応答ヘッダ格納形式,
    res_body     = 応答ボディ格納先変数名,
    res_body_file = 応答ボディ格納先ファイルパス または 変数名,
    res_multipartdata_split = true / false, // マルチパートデータの場合の分割有無
    res_cookie   = 応答クッキー格納変数名,
    res_cookie_file = 応答クッキー格納ファイルパス または 変数名,
    res_cookie_filetype = 応答クッキー格納形式,
    http_version = HTTP バージョン格納変数名,
    reason_phrase = 捕捉メッセージ格納変数名,
    status_code = 結果返却先変数名
)
```

設定項目が多いので、WSS 出力時には選択されなかったり、設定が省略されたりする項目もコメントとして出力している。

メソッド method には次のいずれかを指定する

表 8-68 HTTP2 のメソッド

メソッド
$HTTP.Get
$HTTP.Put
$HTTP.Post
$HTTP.Delete
$HTTP.Patch

メソッド
$HTTP.Head

response_timeout は省略可能。省略されたときには 10000 (10 秒) が指定されたとする。

auth に false を指定したときには、auth_user と auth_password は省略可能。

次の項目はどちらか一方を指定する。両方指定した場合は file の指定を優先する。
特別に要求に含める内容がないときには両方とも省略して構わない。

表 8-69 HTTP2 の req 引数

変数または値からを指定	ファイルからを指定
req_header	req_header_file
req_param	req_param_file
req_cookie	req_cookie_file
req_body	req_body_file

次の項目は応答の当該要素を保存する場合にどちらか一方を指定する。

両方指定した場合は file の指定を優先する。

当該要素を保存しない場合は両方とも省略して構わない。

表 8-70 HTTP2 の res 引数

変数に保存を指定	ファイルに保存を指定
res_header	res_header_file
res_cookie	res_cookie_file
res_body	res_body_file

res_header_filetype と res_cookie_filetype には
$HTTP2.RawFormat または $HTTP2.JSONFormat を指定する。

省略可能で、省略した場合は $HTTP2.RawFormat が指定されたとみなす。

表 8-71 HTTP2 の応答格納形式

応答格納形式	説明
$HTTP2.RawFormat	改行区切りでそのまま格納する
$HTTP2.JSONFormat	JSON 形式で格納する

アップロードするファイルがある場合、fileupload_info と req_fileupload_file のどちらか一方を指定する。

両方指定した場合は req_fileupload_file を優先する。

fileupload_info の構文は次のとおり。ファイル名、フォーム名、コンテントタイプの三つ組みをファイルの個数だけ並べる。

```
fileupload_info = (
    (filename = ファイルパスまたは変数名,
     name = 文字列または変数名,
     content_type = 文字列または変数名),
    (filename = ファイルパスまたは変数名,
     name = 文字列または変数名,
     content_type = 文字列または変数名),
    ...
)
```

fileupload_info の例

```
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 と reason_phrase は省略可能。

メソッド毎に有効な引数（〇）と無視される引数（×）は次のとおり。

表 8-72 HTTP2 のメソッドと引数

引数	メソッド					
	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

引数	メソッド					
auth_pass	○	○	○	○	○	○
req_cookie	○	○	○	○	○	○
req_cookie_file	○	○	○	○	○	○
fileupload_info	×	○	○	×	○	×
res_header_filetype	○	○	○	○	○	○
res_header	○	○	○	○	○	○
res_header_file	○	○	○	○	○	○
res_body	○	○	○	○	○	○
res_body_file	○	○	○	○	○	○
res_multipartdata_split	○	○	○	○	○	○
res_cookie_filetype	○	○	○	○	○	○
res_cookie	○	○	○	○	○	○
res_cookie_file	○	○	○	○	○	○

1. Put, Post, Patch では、パラメータと要求ボディはどちらか一方しか指定できません。

値を status_code で指定した変数に返す。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

詳細や注意事項については、『WinActor ユーザライブラリサンプル説明書』の『HTTP（詳細）』を参照のこと。

8.4.18 JSON 形式書き込み

```
WinActor.JsonWrite プリアンブル
 (
  json_table = JSON 値,
  write_file = ファイルパス,
  variable = 結果返却先変数名,
  use_file = true / false // true のとき write_file を採用する
 )
```

JSON 値は、キー名<型名> = 式 をカンマで区切って並べたリストで表現する。

型名は以下のいずれか。型が string, object, array, null の場合は、値は文字列を与える。

型が真偽値の場合は、true または false を指定する。

表 8-73 JSON 値の型名

型名	説明
integer	整数値
float	小数値
string	文字列
object	オブジェクト
array	配列
boolean	真偽値
null	Null 値

write_file で指定したファイル、または、variable で指定された変数に結果を格納する。

write_file と variable はどちらかを指定する。両方指定された場合、use_file が true なら write_file を、false なら variable を採用する。両方指定されて use_file が省略された場合は、write_file の指定を採用する。

代入や式の中に現れた場合は、結果返却先変数名には結果は格納しない。結果返却先変数名の指定は省略可能。フローチャートの WSS 出力で式（代入文）になったときには結果返却先変数名の指定は省略される。

例:

```
i = 128;
ret = WinActor.JsonWrite [name = "JSON 形式書き込み", 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 }" )
   );
```

実行後の変数 ret の内容

```
"{
  "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 JSON 形式読み取り

JSON オブジェクトが格納された変数、または、ファイルから JSON のキーの値を読み込む。

```
WinActor.JsonRead プリアンブル
    (
        json_table = (キー名 = 変数名, ...),
        read_file = ファイルパス文字列 または ファイルパス文字列が格納された変数名,
        variable = 変数名,
        use_file = true / false // true のとき read_file を採用する
    )
```

read_file で指定されたファイル内容、または、variable で指定された変数に格納された JSON オブジェクトから読み込む。

read_file と variable はどちらかを指定する。両方指定された場合、use_file が true なら read_file を、false なら variable を採用する。両方指定されて use_file が省略された場合は read_file の指定を採用する。

json_table には、JSON オブジェクトから値を読み込むキー名と読み込んだ値を格納する変数名を指定する。

JSON 型名は不要。

8.4.20 その他の JSON ライブラリ

「JSON 形式書き込み」と「JSON 形式読み込み」以外の JSON ライブラリは、スクリプト実行ノードで実装されているので、各スクリプト実行ノードの「注釈」を参照のこと。

8.4.21 Socket 通信

```
WinActor.Socket プリアンブル
  (
    identifier = 変数,                    // コネクション識別子

    connect = true / false,               // 接続機能使用の有無
    host = 文字列 または 変数,            // 接続先
    port = 整数値 または 変数,            // ポート番号 0 ~ 65535
    connect_timeout = 整数値 または 変数,        // 接続タイムアウトミリ秒、-1 以上整数
    connect_exception_name = 文字列 または 変数,         // 接続リトライ例外名

    send = true / false,                  // 送信機能使用の有無
```

```
    send_timeout = 整数値 または 変数,          // 送信タイムアウトミリ秒、0 以上整数
    send_data_type_file = true / false, // データ取得元がファイルか
    send_data_file = 文字列 または 変数,  // データ取得元ファイルパス
    send_data_value = 文字列 または 変数, // 送信データ
    send_conv_text = true / false,       // テキスト変換処理の有無
    send_input_char = 入力文字コード,
    send_input_newline = 入力改行コード,
    send_output_char = 出力文字コード,
    send_output_newline = 出力改行コード,
    send_conv_bin = true / false,        // バイナリ変換（base64）処理の有無
    send_buffer_size = 整数値 または 変数,       // 送信バッファサイズ、0 以上整数
    send_data_size = 変数,               // 送信済データサイズ格納変数
    send_shutdown = true / false,        // 送信シャットダウン処理の有無
    send_exception_name = 文字列 または 変数,    // 送信リトライ例外名,

    receive = true / false,              // 受信機能使用の有無
    receive_timeout = 整数値 または 変数,        // 受信タイムアウトミリ秒、0 以上整数
    receive_data_type_file = true / false,       // データ格納先がファイルか
    receive_data_file = 文字列 または 変数,      // データ格納先ファイルパス
    receive_data_variable = 変数,        // データ格納変数
    receive_conv_text = true / false,    // テキスト変換処理の有無
    receive_input_char = 入力文字コード,
    receive_input_newline =入力改行コード,
    receive_output_char = 出力文字コード,
    receive_output_newline = 出力文字コード,
    receive_conv_bin = true / false,     // バイナリ変換（base64）処理の有無
    receive_buffer_size = 整数値 または 変数,    // 受信バッファサイズ、0 以上整数
    receive_end_condition = 受信終了条件,
    receive_data_size = 整数値 または 変数,              // 受信データサイズ、0 以上整数
    received_data_size = 変数,                   // 受信済みデータサイズ格納変数
    receive_shutdown = true / false,             // 受信シャットダウン処理の有無
    receive_exception_name = 文字列 または 変数,         // 受信リトライ例外名,

    disconnect = true / false,           // 切断機能使用の有無
    disconnect_timeout = 整数値 または 変数,     // 切断タイムアウトミリ秒、-1 以上整数
    disconnect_exception_name = 文字列 または 変数       // 切断リトライ例外名
)
```

使用する機能は connect, send, receive, disconnect で設定する。

使用していない機能の項目も設定可能で、設定した値は保存される。使用していない機能の引数に変数を指定したときには、その変数の宣言が必要である。例えば、send=false, send_data_type_file=true, send_data_value=variable とした場合、送信機能を使わず、送信データもファイルから得る設定だが、variable の宣言が必要となる。

送信機能のデータ取得元を文字列または変数とした場合（send_data_type_file = false とした場合）は、send_input_char に指定できる文字コードは $SOCKET.CharASCII のみ。

受信機能のデータ格納先を変数とした場合（receive_data_type_file = false とした場合）は、receive_output_char に指定できる文字コードは $SOCKET.CharASCII のみ。

コネクション識別子 identifer の引数は必須。

それ以外の項目は引数を省略可能。引数省略時の値は下記のとおり。

表 8-74 引数省略時の値

引数の種類	省略時の値
true / false を指定する引数	false
文字列 または 変数を指定する引数	空文字列
変数を指定する引数	無名識別子
ポート番号	0
タイムアウト	10,000（ミリ秒）
入力・出力バッファサイズ	8,192
入力・出力文字コード	$SOCKET.CharASCII
入力・出力改行コード	$SOCKET.NewLineCRLF
受信終了条件	$SOCKET.EndFin
受信データサイズ	0
リトライ例外名	$SOCKET.ActionException

ただし、条件により引数が必須となる項目がある。

表 8-75 引数が必須となる項目

引数が必須となる項目	引数が必須となる条件
host	connect = true
port	connect = true
send_data_size	send = true
received_data_size	receive = true
receive_data_size	receive_end_condition = $SOCKET.EndSize

send_data_type_file = true の場合 send_data_file に文字列または変数で指定したファイルパスのファイルが送信データの取得元となる。send_data_type_file = false の場合 send_data_value に文字列または変数で指定した値が送信データとなる。

receive_data_type_file = true の場合 receive_data_file に文字列または変数で指定したファイルパスのファイルが受信データの格納先となる。receive_data_type_file = false の場合 receive_data_variable で指定した変数が受信データの格納先となる。

送信テキスト変換処理の有無 send_conv_text と送信バイナリ変換処理の有無 send_conv_bin の両方に true は指定できない。同様に、受信テキスト変換処理の有無 receive_conv_text と受信バイナリ変換処理の有無 receive_conv_bin の両方に true は指定できない。

send_input_char, send_output_char, receive_input_char, receive_output_char に指定する文字コードは下表の既定義定数、IANA 登録名文字列、または、コードページ番号を指定する。コードページ番号は 0 から 65,535 までの整数で指定する。IANA 登録名文字列、または、コードページ番号で指定した場合、実行環境でそのコードがサポートされていない場合は警告を出力し、$SOCKET.CharASCII が指定されたとみなす。

表 8-76 文字コード

既定義定数	備考
$SOCKET.CharASCII	
$SOCKET.CharUTF-16LE	リトルエンディアン BOM なし
$SOCKET.CharUTF-16LE-BOM	リトルエンディアン BOM あり
$SOCKET.CharUTF-16BE	ビッグエンディアン BOM なし
$SOCKET.CharUTF-16BE-BOM	ビッグエンディアン BOM あり
$SOCKET.CharUTF-8	BOM なし
$SOCKET.CharUTF-8-BOM	BOM あり
$SOCKET.CharShift-JIS	コードページ番号 932
$SOCKET.CharEUC-JP	コードページ番号 51932

send_input_newline, send_output_newline, receive_input_newline, receive_output_newline に指定する改行コードは、下表の既定義定数のいずれかを指定する。

表 8-77 改行コード

既定義定数	備考
$SOCKET.NewLineCRLF	
$SOCKET.NewLineCR	
$SOCKET.NewLineLF	
$SOCKET.NewLineNONE	send_output_newline, receive_output_newline に指定したときは、データの末尾の改行コードを取り除いて送信、または、格納する。

receive_end_condition に指定する受信の終了条件は、下表の既定義定数のいずれかを指定する。

表 8-78 受信終了条件

既定義定数	備考
$SOCKET.EndFin	FIN を受信するまで
$SOCKET.EndSize	receive_data_size で指定したオクテット数に到達するまで
$SOCKET.EndEmpty	読み出し可能なデータがなくなるまで

connect_exception_name, send_exception_name, receive_exception_name, disconnect_exception_name に指定するリトライ例外名には $SOCKET.ActionException、または例外名文字列を指定する。$SOCKET.ActionException を指定したときには WSS シナリオを読み込んだ時点の言語設定により、”アクション例外”（日本語）または “ActionException”（英語）が指定されたとみなす。

例外名文字列として空文字列を指定すると例外として扱わずに処理を継続する。

8.5　アダプタアクションに記載のないライブラリ

アダプタアクションに記載のないライブラリは、スクリプト実行ノードやサブルーチンなどで実装されている。

これらのライブラリは、WinActor 本体で WSS 出力可能シナリオに配置して使用することができる。

配置したシナリオをファイルに保存することで、これらのライブラリは wss7 ファイルと uss7 ファイルに保管される。

9. 式復元機能の注意事項

WSS で記述した式を WinActor に読み込むと、WinActor の四則演算ノードで表現するための追加のノードや作業変数が現れる場合がある。このフローチャートをファイルに保存すると、ユーザが記述した式が分解された状態で保存され、スクリプトが見づらくなっていた。

これを軽減するために、スクリプトファイルに保存する時点でできるだけ元の式に復元する機能を導入した。

ただし、フローチャート上で作業変数を再利用するといった編集がされると、元の式に戻したときに意図した動作と異なってしまう場合がある。よって、以下の場合は作業変数をスクリプトに残す。残す場合は、スクリプト全体でその作業変数が残る。

■ 作業変数が残る場合

- 作業変数を参照するノードを追加
- 値の参照元を作業変数に変更
  - 値の設定先: 「値または変数名」「変数名」の箇所全般
- 作業変数に代入するノードを追加
- 値の設定先を作業変数に変更
  - 値の設定先: ノードの値の戻し（結果の格納先）変数
    - スクリプトアダプタの VB からの戻し変数 $...$
    - サブルーチン呼び出しノードの返り値
    - サブルーチン終了ノードの返り値
    - シナリオファイル呼び出しノードの戻り値
    - シナリオファイル終了ノードの戻り値
- 分解で追加されたノードに付箋を付与
- 分解で追加されたノードを breakpoint のターゲットノードとした
- サブルーチン呼び出しで、作業変数に対応する値を「省略」から値や変数を指定した
- サブルーチンの「ローカル変数の設定（終了時に開始時の値を復元する変数)」から作業変数を削除
- シナリオファイル呼び出しの戻り設定に指定した作業変数

■ ユーザが変更していなくても復元されない場合

- while 文と dowhile 文の条件に現れた作業変数
- while 文と dowhile 文の範囲指定の終了（上限）の式に現れた作業変数

- switch 文の case の条件に作業変数が現れて switch-case 文が if 文に変換されている場合、元の swich-case 文に戻らない
- 無名のサブルーチンに現れている作業変数はシナリオ全体で残る
- シナリオファイル呼び出しの値の「呼び出し先から引き継ぐ変数」に呼び出すシナリオで使われている作業変数を指定した場合、呼び出し先シナリオでの復元機能で対象の作業変数が消去されることがある。この場合、引き継ぎは無視される
- 浮きグループをサブルーチンに変換した場合、浮きグループにあった作業変数はシナリオ全体で残る

WinActor® プログラミング言語
WinActor Scenario Script

NTTアドバンステクノロジ株式会社

Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION

本書は著作権法上の保護を受けています。本書の一部あるいは全部を無断で複写、複製することは禁じられています。

本マニュアルの内容は予告なく変更される場合があります。

WA7-U- 20250603