ESSO Scripting Language

- ESSO Scripting Language
- 1. Global functions
- 1.1. ESSO SendKeys syntax
- 2. secretStore object
- 3. SystemInfo object
- 4. Window class
- 5. Document class
- 6. Element class
- 7. Collection class
- 8. File class
- 9. Directory class
- 10. MailService class
- 11. NetworkResource class
- 12. Registry class
- 13. ServerInfo class
- 14. Hll class (version 1.4.0)
- ESSO Scripting examples

ESSO Scripting Language

Introduction

The scripting language used is a full ECMAScript interpreter. Nevertheless, it's not a Javascript interpreter as it's used on web browsers. It only has the core elements of ECMAScript (Objects, Arrays, String, etc.) and a set of objects and functions specially designed for its purpouse.

When the script is run, the elements matched, either window or HTML components, with a ref-as tag will be declared as global variable. Additionally, in the case of web applications, a global variable named document contains the reference to the full HTML document.

Here are the supported functions and classes:

1. Global functions
    1. ESSO SendKeys syntax
2. secretStore object
3. SystemInfo object
4. Window class
5. Document class
6. Element class
7. Collection class
8. File class
9. Directory class
10. MailService class
11. NetworkResource class
12. Registry class
13. ServerInfo class
14. Hll class (version 1.4.0)

1. Global functions

Global functions can be used in an **action** element:

debug	text: string	Sends a message to the debug console.
sleep	millis: int	Stops script execution for the specified milliseconds. Never stops the execution of the application.
env	text: string returns string	Gets the value of an environment variable.
exec	text: string [Dir: string]	Run an external application. Optionally, you can specify a directory to change to.
execWait	text: string [Dir: string]	Executes external application and waits for its completion. Optionally, you can specify a directory to change to.
sendKeys	text: string	Simulates the pressing of the keys indicated. You can see more information on the ESSO sendKeys syntax page.
sendText	text: string	Simulates keystroke typed text. Unlike the SendKeys function, the text is sent, verbatim, without any interpretation.
alert	text: string	Displays a confirmation message to the user.
progress	text: string	Displays a progress message without confirmation from the user.
cancelProgress		Hide progress message.
alertNoWait	text: string	A message, but does not expect the user confirmation.

1.1. ESSO SendKeys syntax

SendKeys syntax

The sendKeys function aims to perform as the user pressing keystrokes. Thus, the function SendKeys ("ABC") simulates to press those three letters.

The keystrokes will be done independently of the application that generates them. Thus, it is possible to press keys that change the focused control or even the active application using sendKeys.

To mimic the action of pressing to keys simultaneously, you can make use of modifiers. The available modifiers are listed at the attached table. So, to send the character ':' you can execute *sendKeys("+.")*. This combination will simulate pressing the Shift key along with '.'. In order to allow a simpler syntax, you can use parentheses to specify more than one key affected by modifiers. Thus, the function SendKeys ("+ (hello)") generates the word HELLO in upper case.

Modifier	**Key**
@	WINDOWS
+	SHIFT
^	CTRL
%	ALT

It should be noted that this method can not get accented letters as a parameter, but the combination necessary to generate the desired letter. This method has a big drawback. The combination of keys needed to get a letter can be different depending of the current keyboard layout. So, in order to send arbitrary text characters, it is recommended to use the function sendText.

Additionally, the sendKeys function supports this “virtual keystrokes” that do not correspond to a specific key but an actual action do be done:

Tag	**Action taken.**
VKEY {X}	The virtual code key (VKEY) will be sent. It is useful to use non-standard keys.
XY {BEEP}	A sound with a frequency X during Y time (in milliseconds).
{DELAY X}	X milliseconds pause.

{DELAY = X}	Make a dealy of X milliseconds between each simulated keystrokes.
{AppActivate WindowTitle}	Bring foreground and activates the application with the specified title.

To send function keys, you can use the following codes:

Key	**Tag**	**Key**	**Tag**
Backspace	{BACKSPACE} or {BS}	F1	{F1}
Break	{BREAK}	F2	{F2}
CapsLock	{CAPSLOC}	F3	{F3}
Delete	{DELETE} or {DEL}	F4	{F4}
Down arrow	{DOWN}	F5	{F5}
End	{END}	F6	{F6}
Enter	{ENTER} or ~	F7	{F7}
Ex	{ESC}	F8	{F8}
Help	{HELP}	F9	{F9}
Home	{HOME}	F10	{F10}
Lnsert	{INS}	F11	{F11}
Left arrow	{LEFT}	F12	{F12}
Num Lock	{NUMLOCK}	F13	{F13}
Next page	{PGDN}	F14	{F14}
Previous page	{PGUP}	F15	{F15}
Print screen	{PRTSC}	F16	{F16}
Right Arrow	{RIGHT}	^	{CARET}
Scroll Lock	{SCROLL}	~	{Tilde}
Tab	{TAB}	{	{LEFTBRACE}
Up arrow	{UP}	}	{RIGHTBRACE}
+ (Numeric keypad)	{ADD}	(	{LEFTPAREN}
- (Numeric keypad)	{SUBSTRCT}	)	{RIGHTPAREN}
* (Numeric keypad)	{Multiply}	windows (left)	{LWIN} or {RWIN}
/ (Numeric keypad)	{DIVIDE}	windows (right)	{RWIN}
+	{PLUS}	context menu	{APPS}
@	{AT}		

Here are a few small examples :

sendKeys parameter	**Effect**
{DELAY=50}	1. Specifies a pause of 50 milliseconds between keystrokes.
@R	2. Click Windows + R to invoke the run dialog command.
notepad~	3. Enter notepad and pressed enter.
hello world!	4. Write "Hello world!".
%ua	5. Click the button to Alt + u, to show the About dialog box ...
{Delay = 100}	1. Specifies a pause of 100 milliseconds between keystrokes.
{AppActivate Calculator}	2. Turn the calculator.
{ESC}	3. Click ESC to clear the contents.
5*7~	4. Write 5 * 7 and pressed Enter.
{beep 1000 500}	5. Makes noise.
^C	6. Press Control-C to copy the contents.
{appactivate Notepad}	7. Switch to notepad.
^E	8. Click to e-control dial all.
{DEL}	9. Click Delete to delete the contents.
Result of 5 * 7 is:	10. Writes "The result is 5 * 7".
^V	11. Click control-v hold the result..
{DELAY=500}{NUMLOCK}{CAPSLOCK}{SCROLL}	1. Turn numbers, uppercase and scroll on in order.
{SCROLL}{CAPSLOCK}{NUMLOCK}	2. Turn them off in the reverse order.
{DELAY=500}	1. Specifies a pause of 500 milliseconds between keystrokes.
%?	2. Click Alt-Space.
{DOWN 5}	3. Click the down arrow five times.

2. secretStore object

Introduction

This object is always visible from any action, and provides access to the user's passwords and secrets. User passwords are always related to a system account.

This is the object used to retrieve user and password in order to **inject credentials** into applications.

Methods

getSecret	text: string returns string	Gets the value of a secret.
getAccounts	system: string returns string []	Gets the list of accounts available for a given systemsecretStore object.
getAccount	system: string returns string	Gets the account to use a particular system. If more than one are available, the system will prompt the user for the one to use. If the user cancels the dialog box, an exception will be thrown. If no account is available, the undefined value will be returned.
getPassword	system: string account: string returns string	Gets the password bound to the account on the system requested.
setPassword	system: string account: string newPassword: string	Changes the password at the password vault (version 1.4)
setSecret	name: string value: string	Sets the value of a secret at the password vault (version 1.4)
generatePassword	system: string account: string returns string	Generates a random password suitable for the selected account (version 1.4)

3. SystemInfo object

Introduction

The SystemInfo object is always visible from any **action**, and provides access to information about the machine.

Attributes

os	string	Specifies the name of the operating system: Windows / Ubuntu
oSVersion	string	Indicates the version of the operating system.
osDistribution	string	Distributor operating system: Microsoft / Ubuntu / RedHat / ....
hostName	string	Return the team name.
clientHostName	string	For remote connections, returns the name of the source host.
fileSeparator	string	Separator files depending on the platform: / Linux \ For Windows
username	string	Operating system user name. It can be different from Soffid user name.

4. Window class

Introduction

When an action is bound with a user interface application, it creates an object of class Window for each component at the XML descriptor with a ref-as attribute. Those components have the following methods:

Methods

getText	returns string	Gets the text value of the component.
setText	text: string	Change the text value of a component.
click		Acts as if the user clicks on the component. It's suitable on button components.
setFocus		Move the focus to the component.

5. Document class

Introduction

When an action is associated with a Web application, it creates a document that identifies the full HTML document. This object assigned to the document variable. Thus, scripts can access the web contents and its DOM tree in runtime. The document object implement a subset of the standard DOM HtmlDocument.

Attributes

url	string	Full URL of the document.
domain	string	Contains the domain of the page.
title	string	Title of the document.
cookie	string	Contains cookies that are bound to this document.
anchors	Collection	Contains elements of type A.
forms	Collection	Contains items of type FORM.
images	Collection	Contains items from IMG.
links	Collection	Contains elements of type A and AREA.
documentElement	Item	Contains the root element.

Methods

getElementById	id: string Element returns	Find the first element with the specified ID.
getElementsByTagName	tag: string returns Collection	Find all elements with the specified tag.
write	text: string	Add content to the document.

writeln	text: string	Add content and new line to the document.
autofill	text: string	Proceed with smart auto fill engine, allowing end user to select and or save accounts. The mandatory parameter sets the domain where to look for accounts.

6. Element class

Introduction

The objects of type Element are created for each input element with a ref-as attribute, or are obtained from the Document itself. It implements a subset of the DOM class HtmlElement.

Attributes

childNodes	Collection	Vector of children elements.
disabled	bool	Indicator whether the element is disabled or not.
id	string	Id attribute of the element.
tagName	string	Element's tag.
parentNode	Item	Parent element.

Methods

getAttribute	name: string returns string	Gets the attribute value of the element.
setAttribute	name: string value: string	Updates the attribute value.
removeAttribute	name: string	Removes an attribute.
getElementsByTagName	tag: string returns Collection	Find all children with the specified tag.
click		Acts as if the user had clicked on the button.
blur		Remove focus.
focus		Grants focus.
submit		Send form contents.

7. Collection class

Introduction

The collection object implements a subset of the standard DOM HTMLCollection

Attributes

length	Long	Number of items in collection.

Methods

item	id: long returns Element	Find the element with id order number. The first element is 0.
namedItem	id: string returns Element	Search for an item with the given name. First search an element with matching Id attribute. If none is found, search for an element with matching Name attribute.

8. File class

Introduction

It allows easy manipulation of files using the File class.

Constructor

File	file: string mode: string	Create an object of type File for the specified file. If mode is “r”, the file will be opened in read mode. If mode is “w”, the file will be opened in write mode. If the mode is “a”, the file will be open in append mode.

Methods

read	byte int (1000) returns string	Reads at most the specified number of bytes. When no numer is specified, 1000 bytes will be read at most.
readLine	returns string	Reads untill end of line.
write	text: string	Writes the specified text.
WriteLine	text: string	Writes text with and end of line.
close		Closes the file.
flush		Flush all buffers to disk.
eof	returns boolean	Returns true if the end of file has been reached.

(static) methods and attributes

Additionally, the File object has the following (static) methods and attributes:

mkdir	directory: string	Creates the specified directory.
stdin	File	Attribute that contains a File object associated with standard input.
stdout	File	Attribute that contains a File object associated with standard output.
stderr	File	Attribute that contains a File object associated with the standard error output.
copy	source: string target: string	Copy selected file. This method is not able to copy directories.
delete	file: string	Deletes a file or directory.
move	source: string target: string	Moves (or renames) a file or directory.
isDirectory	f: string returns boolean	Returns true if the specified file is a directory.
canRead	f: string returns boolean	Returns true if the file can be read.
canWrite	f: string returns boolean	Returns true if the file can be written.
getParent	f: string returns string	Returns the parent directory of a file.

9. Directory class

Introduction

This class is able to look for directories content. A directory object has the following attributes and methods:

Constructor

Directory	file: string	Creates a directory object bound to the specified path.

Methods

length	returns int	Indicates the number of files in the directory.
item	return string	Specifies the name of the file content.

10. MailService class

Introduction

Simple tool to send emails. The MailService object has the following methods.

Constructor

MailService		Create an object of type MailServer.

Methods

setServer	server: string	Specifies the name of the mail server.
setFrom	from: String	Specifies the name of the sender.
Setter	to: string	Specifies the name of the recipient.
send	text: string	Send the message indicated.

11. NetworkResource class

Introduction

Connect and disconnect network services (disks and printers).

Constructor

NetworkResource		Creates an object of type NetworkResource.

Methods

connectPrinter	resource: string model: string	Connects a remote printer to the local spooler.
connectDrive	localDrive: string resource: string password: string (optional) user: string (optional)	Connects a network drive.
disconnectAllPrinters		Disconnects all remote printers.
disconnectPrinter	name: string	Disconnects a remote Printer.
disconnectDrive	localName: string	Disconnects a remote drive.

12. Registry class

Introduction

Manipulate the windows registry.

Constructor

Registry	path: string	Create an object of type Registry

Global objects

Registry.HKEY_LOCAL_MACHINE	Tree Key LOCAL_MACHINE
Registry.HKEY_CURRENT_USER	Tree CURRENT_USER key
Registry.HKEY_USERS	Tree Key USERS
Registry.HKEY_CLASSES_ROOT	Tree Key CLASSES_ROOT
Registry.HKEY_LOCAL_MACHINE32	Tree LOCAL_MACHINE 32-bit keys
Registry.HKEY_CURRENT_USER32	Tree CURRENT_USER key 32bit
Registry.HKEY_USERS32	Tree 32bit key USERS
Registry.HKEY_CLASSES_ROOT32	Tree CLASSES_ROOT 32-bit keys

Methods

openKey	path: string returns Registry	Opens a registry subkey.
createKey	path: string returns Registry	Creates a registry subkey.
getValue	entryName: string Object returns	Reads registry value.

setValue	entryName: string value: Object type: string	Updates a registry value. Type (optional) can be: - REG_SZ - REG_EXPAND_SZ - REG_BINARY - REG_DWORD - REG_MULTI_SZ

13. ServerInfo class

Introduction

This helper class allows the script to query information stored at Soffid console.

Constructor

ServerInfo	path: string	Queried the server returning an object of type ServerInfo.

Methods

length	returns int	Returns the number of rows obtained.
row	n: int object returns	Returns information about the n-throw from.

The objects returned depend on the path indicated.

14. Hll class (version 1.4.0)

Introduction

The Hll class gives the script engine access to Hll terminal emulators. When a hll pattern matches the emulator screen, a hll object of class Hll will be crated and can be used by the action script.

Attributes

sessionId	string	Full URL of the document.
sessionName	string	Contains the domain of the page.
columns	int	Numer of columns.
rows	int	Number of rows.

Methods

getCursorLocation	returns object with row and column attributes	Gets the cursor location.
setCursorLocation	row: integer column: integer	Changes cursor location.
getContent	returns String	Gets the terminal emulator screen content.
sendText	text: string	Send text to host.
sendKeys	keys:string	Send text, possibly containing escape sequences.

Escape sequence

The following escape sequence are defined:

Mnemonic	Meaning	3270	5250	VT
@B	Left Tab	Yes	Yes	No
@C	Clear	Yes	Yes	No
@D	Delete	Yes	Yes	No
@E	Enter	Yes	Yes	No
@F	Erase EOF	Yes	Yes	No
@H	Help	No	Yes	No
@I	Insert	Yes	Yes	No
@J	Jump (Set Focus)	Yes	Yes	No
@L	Cursor Left	Yes	Yes	Yes
@N	New Line	Yes	Yes	Yes
@O	Space	Yes	Yes	Yes
@P	Print	Yes	Yes	Yes
@R	Reset	Yes	Yes	No
@T	Right Tab	Yes	Yes	Yes
@U	Cursor Up	Yes	Yes	Yes
@V	Cursor Down	Yes	Yes	Yes
@X*	DBCS (Reserved)	Yes	Yes	No
@Z	Cursor Right	Yes	Yes	Yes
@0	Home	Yes	Yes	No
@1	PF1/F1	Yes	Yes	No
@2	PF2/F2	Yes	Yes	No
@3	PF3/F3	Yes	Yes	No
@4	PF4/F4	Yes	Yes	No
@5	PF5/F5	Yes	Yes	No
@6	PF6/F6	Yes	Yes	Yes
@7	PF7/F7	Yes	Yes	Yes
@8	PF8/F8	Yes	Yes	Yes
@9	PF9/F9	Yes	Yes	Yes
@a	PF10/F10	Yes	Yes	Yes
@b	PF11/F11	Yes	Yes	Yes
@c	PF12/F12	Yes	Yes	Yes

Mnemonic	Meaning	3270	5250	VT
@d	PF13	Yes	Yes	Yes
@e	PF14	Yes	Yes	Yes
@f	PF15	Yes	Yes	Yes
@g	PF16	Yes	Yes	Yes
@h	PF17	Yes	Yes	Yes
@i	PF18	Yes	Yes	Yes
@j	PF19	Yes	Yes	Yes
@k	PF20	Yes	Yes	Yes
@l	PF21	Yes	Yes	No
@m	PF22	Yes	Yes	No
@n	PF23	Yes	Yes	No
@o	PF24	Yes	Yes	No
@q	End	Yes	Yes	No
@u	Page Up	No	Yes	No
@v	Page Down	No	Yes	No
@x	PA1	Yes	Yes	No
@y	PA2	Yes	Yes	No
@z	PA3	Yes	Yes	No
@A@C	Test	No	Yes	No
@A@D	Word Delete	Yes	Yes	No
@A@E	Field Exit	Yes	Yes	No
@A@F	Erase Input	Yes	Yes	No
@A@H	System Request	Yes	Yes	No
@A@I	Insert Toggle	Yes	Yes	No
@A@J	Cursor Select	Yes	Yes	No
@A@L	Cursor Left Fast	Yes	Yes	No
@A@Q	Attention	Yes	Yes	No
@A@R	Device Cancel (Cancels Print Presentation Space)	Yes	Yes	No
@A@T	Print Presentation Space	Yes	Yes	Yes
@A@U	Cursor Up Fast	Yes	Yes	No

Mnemonic	Meaning	3270	5250	VT
@A@V	Cursor Down Fast	Yes	Yes	No
@A@Z	Cursor Right Fast	Yes	Yes	No
@A@9	Reverse Video	Yes	Yes	No
@A@b	Underscore	Yes	No	No
@A@c	Reset Reverse Video	Yes	No	No
@A@d	Red	Yes	No	No
@A@e	Pink	Yes	No	No
@A@f	Green	Yes	No	No
@A@g	Yellow	Yes	No	No
@A@h	Blue	Yes	No	No
@A@i	Turquoise	Yes	No	No
@A@j	White	Yes	No	No
@A@l	Reset Host Colors	Yes	No	No
@A@t	Print (Personal Computer)	Yes	Yes	No
@A@y	Forward Word Tab	Yes	Yes	No
@A@z	Backward Word Tab	Yes	Yes	No
@A@-	Field -	No	Yes	No
@A@+	Field +	No	Yes	No
@A@<	Record Backspace	No	Yes	No
@S@E	Print Presentation Space on Host	No	Yes	No
@S@x	Dup	Yes	Yes	No
@S@y	Field Mark	Yes	Yes	No
@W@C	Edit Copy	Yes	Yes	Yes
@W@D	Edit Clear	Yes	Yes	Yes
@W@E	Edit Copy Append	Yes	Yes	Yes
@W@L	Edit Copy Link	Yes	Yes	Yes
@W@N	Edit Paste Next	Yes	Yes	Yes
@W@V	Edit Paste	Yes	Yes	Yes
@W@X	Edit Cut	Yes	Yes	Yes
@W@Z	Edit Undo	Yes	Yes	Yes

Mnemonic	Meaning	3270	5250	VT
@X@1	Display SO/SI	Yes	Yes	No
@X@5	Generate SO/SI	No	Yes	No
@X@6	Display Attribute	No	Yes	No
@X@7	Forward Character	No	Yes	No
@X@c	Split Vertical Bar	No	Yes	No
@M@0	VT Numeric Pad 0	No	No	Yes
@M@1	VT Numeric Pad 1	No	No	Yes
@M@2	VT Numeric Pad 2	No	No	Yes
@M@3	VT Numeric Pad 3	No	No	Yes
@M@4	VT Numeric Pad 4	No	No	Yes
@M@5	VT Numeric Pad 5	No	No	Yes
@M@6	VT Numeric Pad 6	No	No	Yes
@M@7	VT Numeric Pad 7	No	No	Yes
@M@8	VT Numeric Pad 8	No	No	Yes
@M@9	VT Numeric Pad 9	No	No	Yes
@M@-	VT Numeric Pad -	No	No	Yes
@M@,	VT Numeric Pad ,	No	No	Yes
@M@.	VT Numeric Pad .	No	No	Yes
@M@e	VT Numeric Pad Enter	No	No	Yes
@M@f	VT Edit Find	No	No	Yes
@M@i	VT Edit Insert	No	No	Yes
@M@r	VT Edit Remove	No	No	Yes
@M@s	VT Edit Select	No	No	Yes
@M@p	VT Edit Previous Screen	No	No	Yes
@M@n	VT Edit Next Screen	No	No	Yes
@M@a	VT PF1	No	No	Yes
@M@b	VT PF2	No	No	Yes
@M@c	VT PF3	No	No	Yes
@M@d	VT PF4	No	No	Yes
@M@h	VT HOld Screen	No	No	Yes
@M@(space)	Control Code NUL	No	No	Yes

Mnemonic	Meaning	3270	5250	VT
@M@A	Control Code SOH	No	No	Yes
@M@B	Control Code STX	No	No	Yes
@M@C	Control Code ETX	No	No	Yes
@M@D	Control Code EOT	No	No	Yes
@M@E	Control Code ENQ	No	No	Yes
@M@F	Control Code ACK	No	No	Yes
@M@G	Control Code BEL	No	No	Yes
@M@H	Control Code BS	No	No	Yes
@M@I	Control Code HT	No	No	Yes
@M@J	Control Code LF	No	No	Yes
@M@K	Control Code VT	No	No	Yes
@M@L	Control Code FF	No	No	Yes
@M@M	Control Code CR	No	No	Yes
@M@N	Control Code SO	No	No	Yes
@M@O	Control Code SI	No	No	Yes
@M@P	Control Code DLE	No	No	Yes
@M@Q	Control Code DC1	No	No	Yes
@M@R	Control Code DC2	No	No	Yes
@M@S	Control Code DC3	No	No	Yes
@M@T	Control Code DC4	No	No	Yes
@M@U	Control Code NAK	No	No	Yes
@M@V	Control Code SYN	No	No	Yes
@M@W	Control Code ETB	No	No	Yes
@M@X	Control Code CAN	No	No	Yes
@M@Y	Control Code EM	No	No	Yes
@M@Z	Control Code SUB	No	No	Yes
@M@u	Control Code ESC	No	No	Yes
@M@v	Control Code FS	No	No	Yes
@M@w	Control Code GS	No	No	Yes
@M@x	Control Code RS	No	No	Yes
@M@y	Control Code US	No	No	Yes
@M@z	Control Code DEL	No	No	Yes

Mnemonic	Meaning	3270	5250	VT
@Q@A	VT User Defined Key 6	No	No	Yes
@Q@B	VT User Defined Key 7	No	No	Yes
@Q@C	VT User Defined Key 8	No	No	Yes
@Q@D	VT User Defined Key 9	No	No	Yes
@Q@E	VT User Defined Key 10	No	No	Yes
@Q@F	VT User Defined Key 11	No	No	Yes
@Q@G	VT User Defined Key 12	No	No	Yes
@Q@H	VT User Defined Key 13	No	No	Yes
@Q@I	VT User Defined Key 14	No	No	Yes
@Q@J	VT User Defined Key 15	No	No	Yes
@Q@K	VT User Defined Key 16	No	No	Yes
@Q@L	VT User Defined Key 17	No	No	Yes
@Q@M	VT User Defined Key 18	No	No	Yes
@Q@N	VT User Defined Key 19	No	No	Yes
@Q@0	VT User Defined Key 20	No	No	Yes
@Q@a	VT Backtab	No	No	Yes
@Q@r	VT Clear Page	No	No	Yes
@Q@s	VT Edit	No	No	Yes
@@	@	Yes	Yes	Yes
@$	Alternate Cursor (The Presentation Manager Interface only)	Yes	Yes	Yes
@<	Backspace	Yes	Yes	Yes

Mnemonic	Meaning	3270	5250	VT
@:@s	Screen Reverse	Yes	Yes	Yes
@:@n	Bidi Layer	Yes	Yes	Yes
@:@l	Latin Layer	Yes	Yes	Yes
@:@F	Field Reverse	Yes	Yes	No
@:@p	Push	Yes	No	No
@:@e	End Push	Yes	No	No
@:@a	Auto Push	Yes	No	No
@:@r	Auto Reverse	Yes	No	No
@:@d	CSD	Yes	No	No
@:@f	Final	Yes	No	No
@:@i	Isolated	Yes	No	No
@:@m	Middle	Yes	No	No
@:@t	Initial	Yes	No	No
@:@h	Field Shape	Yes	No	No
@:@u	Field Base	Yes	No	No
@:@b	Base	No	Yes	No
@:@o	Close	No	Yes	No
@:@K	Column Heading	No	No	Yes
@:@B	Cursor Direction	No	No	Yes
@:@D	Encoding Mode	No	No	Yes
@:@M	VT Change Display Mode	No	No	Yes (Hebrew only)

ESSO Scripting examples

1. Run an application like notepad

```
exec ("notepad.exe");
```

2. Automatic application update

```
if (SystemInfo.os == "Linux") {
exec ("(mkdir /tmp/google-chrome-updates && " +
" wget -O /tmp/google-chrome-updates/google-chrome-stable_current_amd64.deb " +
" -c https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb " +
" && apt-get update && sudo apt-get install libappindicator1 " +
" && sudo dpkg -i /tmp/google-chrome-updates/google-chrome-stable_current_amd64.deb " +
" && sudo apt-get install google-chrome-stable) " +
" </dev/null >>/var/log/google-chrome-stable_current_amd641.log 2>&1 "
);
}
```

3.

```
<Mazinger>
  <WebApplication url="https://jira.soffid.com/.*"  >
    <Input id="login-form-username" ref-as="u"/>
    <Input id="login-form-password" ref-as="p"/>
    <Input id="login" ref-as="b"/>
    <Action event="onLoad" type="script" repeat="true" delay="5">
      account = secretStore.getAccount("soffid.org-ldap");
      debug("Account = "+account);
      u.setAttribute("value", account);
      password = secretStore.getPassword("soffid.org-ldap", account);
      debug("Password = "+password);
      p.setAttribute("value", password);
      sleep(100);
      debug("Clicking");
      b.click();
      debug("Clicked");
```

```
    </Action>
  </WebApplication>
</Mazinger>
```