Configuration

Configuration of the initial settings of the utility are set by simply running the helicon executable without parameters, a window with the available configuration settings will be shown.

There might be cases when the application fails to show the configuration window, in most cases that happens because of a corrupted configuration file possibly from previous versions of the utility. Simply check the *log.txt* file on the helicon folder to confirm that is the issue, in which case simply deleting the *config.dat* file would solve the problem.

Variables

All variables in the context can be accessed by using the syntax [[VarName]] where “VarName" is the variable you’d like to access. Also, there are currently the following modifiers supported (note that modifiers are ***case-sensitive***):

Modifier	**Example**	**Description**
STRING	[[STRING varname]]	Converts the specified variable to a string, this is useful for cases when the source is not exactly a string but some other object type.
HEXSTR	[[HEXSTR varname]]	Converts the given variable to a Hex String starting with 0x (like 0x123554784) which is useful to store blob data in a database table.
ESCAPE	[[ESCAPE varname]]	Escapes the specified variable and encloses it within single-quotes. Useful to store strings in a database table.
FILE_EXISTS	[[FILE_EXISTS varname]]	Evaluates to 1 if the file specified in the variable exists, or 0 if not.
FILE_SIZE	[[FILE_SIZE varname]]	Evaluates to the size of the filename specified in the variable.
ENV	[[ENV Path]]	Returns the value of an environment variable.

SELF_PATH	[[SELF_PATH]]	Returns the full path of the HELICON executable (including trailing back-slash).
REGEX_MATCH REGEX_MATCH_CS	[[REGEX_MATCH P T]]	Performs a regex match verification of the given pattern P and the string T, returns “1" if match, or “0" otherwise. The input T will be broken by \n and trimmed. The _CS variant is case-sensitive.
REGEX_MATCH_ML	[[REGEX_MATCH_ML P T]]	Same as REGEX_MATCH but it does not split the input string T and enables multi-line options.
COUNT	[[COUNT ARR]]	Returns the number of elements in the specified array.
STRLEN	[[STRLEN XX]]	Returns the length of a string.
TRIM	[[TRIM XX]]	Trims the white-space off the left and right of the string.
DIR_EXISTS	[[DIR_EXISTS varname]]	Evaluates to 1 if the directory specified in the variable exists, or 0 if not.
UUID	[[UUID]]	Returns a UUID string.
TEMPNAM	[[TEMPNAM]]	Returns a full path to a unique temporal filename.
DIRNAME	[[DIRNAME xxx]]	Returns the full parent directory path of a given file path.
FILENAME	[[FILENAME xxx]]	Returns the filename only (without extension) of a file given its path.

When a tab, space or new-line character is desired, you can use the built-in variables: [[TAB]] [[SP]] and [[NL]] respectively.

To change the text color in ANSI-compatible terminals (Windows 10) you can use the build-in variables: @DEF, @RED, @GREEN, @YELLOW, @BLUE, @MAGENTA and @CYAN before the text of which you want to change the color.

Brief Version History

Version 2.3

As of version 2.3 the string formatter now allows nested modifiers to be used, that means - for instance- that now you can use [[ESCAPE [[FSIZE varname]]]] and such formats without issues.

Version 2.4

RegexExtract, [[Env X]] and SetEnv have been added. Added [[SELF_PATH]], added new element [[FileDelete]].

Version 2.5

- Added PdfLoadTextArray to load plain text contents (no OCR whatsoever) from a PDF into an array (one array element per page).
- Added SplitText used to split a text variable and store it in an array.
- Added REGEX_MATCH and REGEX_MATCH_NL modifiers.
- Added Switch/Case element.
- Added new global attribute "When" to execute an action only if the condition is true.
- Added option "Single" to RegexExtract to just get a single match.
- Added STRLEN, COUNT modifiers and Eval attribute to SetVar.

Version 2.6 / 2.65

- Added TRIM.
- Added Subroutine, Call, CallSubroutine, IFilterLoadText.
- Added "Timeout" to <SqlOpen/>
- Changed semantics of <Stop/> for Subroutines.

Version 2.68

- Added PdfLoadText and included Poppler in the package.

Version 2.71

- Added FileCopy and FileMove.

Version 2.72

- Added SendMail.

Version 2.74

- Added DIR_EXISTS.

Version 2.75

- Added “ResponseType” option to <ApiCall>.

Version 2.76

- Added ImapOpen, ImapClose, ImapSetSeen and modified behavior of ImapLoadArray to make use of globally open Imap connection.

Version 2.82

- Added variables [[UUID]] and [[TEMPNAM]].
- Added action PdfMerge and Sleep.

Version 2.83

- Added [[REGEX_MATCH_CS]]
- Added attribute IgnoreCase to <RegexExtract>
- Added <ReplaceText>

Version 2.84

- Added [[@DEF]], [[@RED]], [[@GREEN]], [[@YELLOW]], [[@BLUE]], [[@MAGENTA]], [[@CYAN]]
- Added <MsgLoadInfo>
- Changed behavior of <FileSave> to automatically create any required directories.
- Added “Recursive” option to ForEachFile.
- Added [[DIRNAME]] and [[FILENAME]]
- Changed behavior of <SendMail> to allow multiple attachments to be specified also using “|” in a single element.

Issues

When running Helicon from the Windows Task Scheduler, if you require to run another program that uses DCOM (like for example Microsoft Word to convert a docx to pdf) you might run into a problem that causes the DCOM not to load. To fix this issue you have to run “Component Services”, select “Computers”, “My Computer” then “DCOM Config” and find the AppID you wish to configure (either by name or by its CLSID), then do a right-click, select properties, select tab “Identity” and then “This User”, you will be prompted with the username and password of the user you want to use to run the task, put the credentials there and click on “Ok”. That should be all.

Service Utility

Helicon comes with a “run as a service” utility named “servx”, in order to install any xml configuration process to run periodically as a service you have to run the following commands (as administrator):

1. `servx install XXX "./my_config.xml"`
   Used to install a service named “XXX” and to ensure it runs “my_config.xml” (which is on the same folder as Helicon and servx due to the “./”) ***infinitely*** while the service is running.
   By default the service will be stopped and its startup mode will be set to “manual”, if you wish to change this startup to automatic such that the service will run with windows startup then open services.msc and configure the startup mode from there.

2. `servx uninstall XXX`
   Used to uninstall a previously installed service named “XXX”. If the service is running it will be automatically stopped.

3. `sc start XXX`
   Used to start a previously installed service named “XXX”, after executing this command you can assume Helicon is running and executing the xml process you specified when installing the service (i.e. my_config.xml).

4. `sc stop XXX`
   Used to stop a service named “XXX”.

5. `servx config XXX "my_config2.xml"`
   Used to set the process configuration xml that will be run by Helicon. If you have a service already installed and you wish to change which xml file it runs you can use this command.

NOTE: Remember to use full path to the XML file when specifying it to servx install or servx config or use the “./” path to indicate your file is in the same folder as Helicon and servx.

♦ SqlOpen

Opens the internal SQL connection to the SQL Server specified in the initial configuration. If any error occurs an exception will be thrown.

Attributes
Timeout Optional attribute to override the default connection timeout in seconds (300).

```
<SqlOpen Timeout="30"/>
```

♦ SqlClose

Closes the internal SQL connection. Recommended for completion purposes, but not required since it will be automatically executed when the process finishes.

```
<SqlClose/>
```

♦ ForEachFile

Executes the actions inside the element for each of the files in a specified directory.

Any error caused when opening the directory will cause an exception to be thrown, however any error occurring inside the foreach-loop (meaning, any error occurring with the actions specified) will simply cause the error to be logged and the execution to resume with the next file.

Attributes

InDirectory
Specifies the directory path to use as source. If none specified the current directory will be used.

WithExtension
(Optional) Space-separated list of extensions (including the leading dot) to filter the files. If none specified no filtering will be applied.

WithMinSize
(Optional) Integer indicating the minimum size (in bytes) of the allowed files. If none specified no filtering will be applied.

WithMaxSize
(Optional) Integer indicating the maximum size (in bytes) of the allowed files. If none specified no filtering will be applied.

IncludeHidden
(Optional) Boolean value (TRUE or FALSE) indicating if hidden files should also be reported. Defaults to false.

Recursive
(Optional) Boolean value (TRUE or FALSE) indicating if the search should be done also in all sub-directories (defaults to false).

Introduced Context Variables

File.Ext
Extension of the file including the leading dot.

File.Name
Name of the file (including extension).

File.Size
Size of the file in bytes.

File.Path
Path to the file's directory, that is the path without the filename, just the directory path without the trailing back-slash (\).

File.FullPath
Full path to the file, that includes the directory and the filename.

```
<ForEachFile InDirectory="C:\" WithExtension=".txt .doc .pdf">
    <Echo>[[File.Name]]</Echo>
</ForEachFile>
```

♦ **SqlStatement**

Executes an SQL statement (innerText). If any error occurs an exception will be thrown, if this behavior is not desired use the Silent attribute. If an empty query is specified only a warning will be issued.

Attributes
Silent When set to TRUE, causes any error during the statement execution to be ignored and written to the log file instead of throwing the exception. Defaults to FALSE.

```
<SqlStatement Silent="TRUE">
    DROP TABLE Test
</SqlStatement>
```

♦ **SqlLoadRow**

Executes a data retrieval query (innerText) and loads the first row (regardless of how many other rows there might be) from the result into the context. It is recommended to use a LIMIT clause to ensure only one row is retrieved.

All column in the row will be available in the context with their respective name, although column names can be prefixed by using the attribute “Prefix" (see below).

Upon any error an exception will be thrown, to override this behavior (not recommended, but sometimes applicable) use the Silent attribute. – If an empty query is specified only a warning will be issued.

Attributes
Silent When set to TRUE, causes any error during the query execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE. **Prefix** Specifies a prefix to be used on all columns. Used to disambiguate when more than one SqlLoadRow action is performed. All columns retrieved will have this prefix prepended to them separated by a dot. See example below.

```
<SqlLoadRow Prefix="X">
```

```
    SELECT name, last_name FROM persons WHERE id=4
</SqlStatement>
```

The above will introduce variables: `X.name` and `X.last_name` into the context.

♦ SqlLoadArray

Executes a data retrieval query (innerText) and loads all rows from the result as an array-variable in the context. If an empty query is specified a warning will be issued.

All column in the row will be available in the context with their respective name, although column names can be prefixed by using the attribute “Prefix" (see below).

Upon any error an exception will be thrown, to override this behavior use the Silent attribute.

Attributes
Silent When set to TRUE, causes any error during the query execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE. **Into** Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow.

```
<SqlLoadArray Into="MyResultSet">
    SELECT name, last_name FROM persons
</SqlLoadArray>
```

♦ ForEachRow

Executes the actions inside the element for each of the rows in the specified array-variable. The contents of each row will be loaded into the context with the respective name of each field (unless a prefix is specified, see prefix attribute below).

Any errors occurring during the for-each loop will not cause the loop to stop, instead the error will be logged, and the for-each loop will continue for the next row. However any error occurring while loading the array-varible will cause exceptions.

Attributes

In
Specifies the name of the array-variable to use as source. If the variable is not an array-variable an exception will be thrown.

Prefix
Specifies a prefix to be used on all introduced variables. When specified all fields loaded from rows will end up in the context having the name “X.Y" where X is the prefix and Y the field name.

```
<SqlLoadArray Into="MyResultSet">
    SELECT name, last_name FROM persons
</SqlLoadArray>

<ForEachRow In="MyResultSet">
    <Echo>Hello [[name]] [[last_name]]</Echo>
</ForEachRow>
```

♦ SafeBlock

Executes the actions inside the **Execute** element, if any exceptions are thrown during the process the actions inside the **OnFailure** element will be executed, when the later happens a new variable named “ERRSTR" will be introduced in the context with the error string of the exception detected.

If no error happens the actions inside the **OnSuccess** element will be executed. This element is optional.

If the OnException element is not provided the SafeBlock will simply write the error to the log and ignore the exception, the inner process sequence (the Execute actions) will be interrupted, of course.

If yet another exception occurs while processing the OnFailure or onSuccess blocks any errors will be written to the log.

```
<SafeBlock>
     <Execute>
         <RaiseException>Testing Exception</RaiseException>
     </Execute>

     <OnFailure>
         <Echo>An error occurred: [[ERRSTR]]</Echo>
     </OnFailure>

     <OnSuccess>
         <Echo>All Good!</Echo>
     </OnSuccess>

</SafeBlock>
```

♦ If

Executes the actions inside the **True** element, if the condition evaluated yields true, otherwise the **False** element will be executed.

The condition attribute can make use of any of the context variables available. Note that special care must be taken when writing conditions because they are evaluated in C#. - If there was any error while evaluating the condition, then the **Error** element will be executed, and when that happens a new variable named “ERRSTR" will be introduced in the context with the error string of the exception detected.

All of the three (**True**, **False** and **Error**) elements are optional.

```
<If Condition="'[[Response]]' != 'OK'">
     <True>
         <RaiseException>Oh no!</RaiseException>
     </True>

     <False>
         <Echo>All Good.</Echo>
     </False>
</If>
```

♦ ApiCall

Executes a call to a remote API function that returns JSON data and stores the results in the context. All elements in the inner area will be sent as POST parameters.

Attributes
Url Address of the API function to call.
Prefix Specifies a prefix to be prepended on all introduced variables. When set all variables will have the name “X.Y" where “X" is the prefix and “Y" is the variable name. (Default prefix is “Api").
ClearCookies Boolean value (TRUE/FALSE) indicating if the cookies previously obtained should be cleared or preserved. Useful when having to perform several function calls and cookies must be preserved. (Default is FALSE).
ResponseType Indicates the type of response you’re expecting. Valid options are RAW and JSON. - Defaults to “JSON”. When “RAW” is selected the output

variable will be “raw” and “rawBytes” (would end up as Api.raw and Api.rawBytes depending on prefix).

- **Regular API Call:**

```
<ApiCall Url="http://someapi.com/api">
    <username>admin</username>
    <password>helloworld</password>
</ApiCall>
```

- **POST a file with inline data:**

```
<ApiCall Url="http://someapi.com/api">
    <file Filename="FakeFilename.txt">
    <![CDATA[Hello! This is the file data to be posted!]]>
    </file>
</ApiCall>
```

- **POST a file from an actual file:**

```
<ApiCall Url="http://someapi.com/api">
<file Filename="FakeFilename.txt" FromFile="Path/To/File.txt"/>
</ApiCall>
```

♦ Pop3LoadArray

Executes data retrieval from a POP3 server and loads results as an array-variable in the context. A maximum number of records to retrieve must be specified.

All fields in the email record will be available in the context with their respective name, although column names can be prefixed by using the attribute “Prefix" (see below).

Upon any error an exception will be thrown, to override this behavior use the Silent attribute.

Attributes
Silent When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE.
Into Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow.

Count
Maximum number of records to process, defaults to 100.

Host
Host address of the POP3 server.

Port
Port of the POP3 server.

SSL
Boolean value (TRUE/FALSE) indicating if SSL should be used.

Username
Name of the user to use to login to the POP3 server.

Password
Password of the user to login to the POP3 server.

The fields available in the output array are:

```
MSG_ID          STRING (UUID of Message)
MSG_DATE        DATETIME (yyyy-MM-dd HH:mm:ss)
MSG_FROM_NAME   STRING
MSG_FROM_EMAIL  STRING
MSG_SUBJECT     STRING
MSG_BODY        STRING
MSG_BODY_LEN    STRING (Length of Body)
```

```
<Pop3LoadArray Into="MyResultSet" Host="MyServer.com" Port="X"
SSL="TRUE" Username="Me" Password="Something" />
```

♦ ImapOpen

Open an IMAP connection for subsequent operations. Useful when you want to mark messages as seen/unseen using a custom process.

Attributes

Silent
When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE.

Host
Host address of the IMAP server.

Port

Port of the IMAP server. **SSL** Boolean value (TRUE/FALSE) indicating if SSL should be used. **Username** Name of the user to use to login to the IMAP server. **Password** Password of the user to login to the IMAP server.

```
<ImapOpen Host="MyServer.com" Port="X" SSL="TRUE" Username="Me"
Password="Something"/>
```

♦ ImapClose

Closes a previously open IMAP connection.

Attributes
Silent When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE.

```
<ImapClose/>
```

♦ ImapSetSeen

Sets the SEEN flag of a message via IMAP protocol. The credentials can be set on this element or they can be left blank -in which case- the global IMAP connection opened with ImapOpen will be used.

Attributes
Silent When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE. **Host** Host address of the IMAP server. **Port**

Port of the IMAP server. **SSL** Boolean value (TRUE/FALSE) indicating if SSL should be used. **Username** Name of the user to use to login to the IMAP server. **Password** Password of the user to login to the IMAP server. **Id** ID of the message to modify. Should match the MSG_ID obtained when retrieving messages. **Value** Boolean value for the SEEN flag (True or False) default is True.

```
<ImapSetSeen Id=”1” Value=”True”/>
```

♦ **ImapLoadArray**

Executes data retrieval from an IMAP server and loads results as an array-variable in the context. A maximum number of records to retrieve must be specified.

All fields in the email record will be available in the context with their respective name, although column names can be prefixed by using the attribute “Prefix" (see below).

Upon any error an exception will be thrown, to override this behavior use the Silent attribute. An IMAP query must be specified in the innerText of the element, possible values are below after the attributes table.

Note that credentials can be left blank, in which case the global IMAP connection opened with ImapOpen will be used.

Attributes
Silent When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE. **Into** Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow. **Count** Maximum number of records to process, defaults to 100.

Host
Host address of the IMAP server.

Port
Port of the IMAP server.

SSL
Boolean value (TRUE/FALSE) indicating if SSL should be used.

Username
Name of the user to use to login to the IMAP server.

Password
Password of the user to login to the IMAP server.

MarkAsSeen
All retrieved emails will be marked as “Seen" afterwards, this is useful to fetch only unseen emails.

SaveEML
Saves the contents of the message as EML and provides a new variable MSG_EML_DATA with the data contents stored as string (can be used with FileSave or HexStr).

The fields available in the output array are:

```
MSG_ID              STRING (UUID of Message)
MSG_UID_VALIDITY    STRING (UID Validity of the Folder)
MSG_DATE            DATETIME (yyyy-MM-dd HH:mm:ss)
MSG_FROM_NAME       STRING
MSG_FROM_EMAIL      STRING
MSG_SUBJECT         STRING
MSG_BODY            STRING
MSG_BODY_LEN        STRING (Length of Body)
MSG_ATTACHMENTS     STRING (Semi-colon separated list of names)
```

Key	Meaning
<sequence set>	Messages with message sequence numbers corresponding to the specified message sequence number set. Examples: "1", "589, 1067", "1:*", "5,7,10:25,37:*", "3:4". Multiple values are delimited with commas (,). Colon (:) is used to specify a range. For instance, "1:10" designates message indices in the range from 1 to 10. "*" is a wildcard. When used in a range, sets higher boundary of the range to the index of the last message in the folder.
ALL	All messages in the folder; the default initial key for ANDing.
ANSWERED	Messages with the \Answered flag set.
BCC <string>	Messages that contain the specified string in the envelope structure's BCC field.
BEFORE <date>	Messages whose internal date (disregarding time and timezone) is earlier than the specified date.
BODY <string>	Messages that contain the specified string in the body of the message.
CC <string>	Messages that contain the specified string in the envelope structure's CC field.
DELETED	Messages with the \Deleted flag set.

DRAFT	Messages with the \Draft flag set.
FLAGGED	Messages with the \Flagged flag set.
FROM <string>	Messages that contain the specified string in the envelope structure's FROM field.
HEADER <field-name> <string>	Messages that have a header with the specified field-name (as defined in [RFC-2822]) and that contains the specified string in the text of the header (what comes after the colon). If the string to search is zero-length, this matches all messages that have a header line with the specified field-name regardless of the contents.
KEYWORD <flag>	Messages with the specified keyword flag set.
LARGER <n>	Messages with an [RFC-2822] size larger than the specified number of octets.
NEW	Messages that have the \Recent flag set but not the \Seen flag. This is functionally equivalent to "(RECENT UNSEEN)".
NOT <search-key>	Messages that do not match the specified search key.
OLD	Messages that do not have the \Recent flag set. This is functionally equivalent to "NOT RECENT" (as opposed to "NOT NEW").
ON <date>	Messages whose internal date (disregarding time and timezone) is within the specified date.
OR <search-key1> <search-key2>	Messages that match either search key.
RECENT	Messages that have the \Recent flag set.
SEEN	Messages that have the \Seen flag set.
SENTBEFORE <date>	Messages whose [RFC-2822] Date: header (disregarding time and timezone) is earlier than the specified date.
SENTON <date>	Messages whose [RFC-2822] Date: header (disregarding time and timezone) is within the specified date.
SENTSINCE <date>	Messages whose [RFC-2822] Date: header (disregarding time and timezone) is within or later than the specified date.
SINCE <date>	Messages whose internal date (disregarding time and timezone) is within or later than the specified date.
SMALLER <n>	Messages with an [RFC-2822] size smaller than the specified number of octets.
SUBJECT <string>	Messages that contain the specified string in the envelope structure's SUBJECT field.
TEXT <string>	Messages that contain the specified string in the header or body of the message.
TO <string>	Messages that contain the specified string in the envelope structure's TO field.
UID <sequence set>	Messages with unique identifiers corresponding to the specified unique identifier set. Sequence set ranges are permitted.
UNANSWERED	Messages that do not have the \Answered flag set.
UNDELETED	Messages that do not have the \Deleted flag set.
UNDRAFT	Messages that do not have the \Draft flag set.
UNFLAGGED	Messages that do not have the \Flagged flag set.
UNKEYWORD <flag>	Messages that do not have the specified keyword flag set.
UNSEEN	Messages that do not have the \Seen flag set.

```
<ImapLoadArray Into="MyResultSet" Host="MyServer.com" Port="X"
SSL="TRUE" Username="Me" Password="Something" MarkAsSeen="TRUE">
SINCE 15-02-2018
</ImapLoadArray>
```

♦ PdfLoadInfo

Loads the information of a PDF file into the context.

Attributes

Path Specifies the path to the file. If the file does not exist no error will be thrown, instead empty varibles will be introduced. (Unless Strict is set). **Prefix** Specifies a prefix to be prepended on all introduced variables. When set all variables will have the name “X.Y" where “X" is the prefix and “Y" is the variable name. **Strict** Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception will be thrown. Same will happen if an empty path is specified. Defaults to FALSE.

Introduced Context Variables
PdfNumPages Number of pages in the PDF file. **PdfMinWidth and PdfMaxWidth** Minimum and maximum page width of the PDF (in inches). **PdfMinHeight and PdfMaxHeight** Minimum and maximum page height of the PDF (in inches).

```
<PdfLoadInfo Path="Info.txt"/>
```

♦ PdfLoadTextArray

Loads the plaint-text contents from a PDF file into the context as an array of objects, each containing two properties “PageNumber" and “PageText".

Attributes
Path Specifies the path to the file. If the file does not exist no error will be thrown, instead empty varibles will be introduced. (Unless Strict is set). **Into** Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow. **Strict**

Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception will be thrown. Same will happen if an empty path is specified. Defaults to FALSE.

```
<PdfLoadTextArray Path="Info.txt" Into="X"/>
```

♦ RegexExtract

Runs a regular-expression extraction on a given text and produces an array that can be used with ForEachRow. Upon any error an exception will be thrown, to override this behavior use the Silent attribute. The regular expression is specified in the innerText of the element. The matching will be done case-insensitive.

If the “Single” attribute is set to true, no array will be returned and instead the REGEX_* variables of a single match will be added to the context.

Attributes
Silent When set to TRUE, causes any error during the execution to be ignored and written to the log file instead of throwing an exception. Defaults to FALSE. **IgnoreCase** Indicates if match should ignore case of the text. Defaults to TRUE. **Single** Indicate to not return an array but only the REGEX_* variables of the first match in the context. **Into** Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow. **Source** Source text to be used with the pattern. One field named **REGEX_N** will be created on the context indicating the number of matches found. The fields available in the output array are the groups of the regular expression:

```
REGEX_0        STRING (Contains the whole match)
REGEX_1        STRING (First group)
REGEX_2        STRING (Second group)
REGEX_I        STRING (Ith Group)
```

```
<RegexExtract Into="MyResultSet" Source="[[InputData]]">
[A-Za-z0-9_.]+@[A_Za-z0-9_]+(\.[A-Za-z0-9_]+)+
</RegexExtract>

<If Condition="[[REGEX_N]] != 0">
<True>
     <ForEachRow>
          <Echo>A : [[REGEX_0]]</Echo>
     </ForEachRow>
</True>
</If>
```

♦ SplitText

Splits the given text (inner contents) by the specified delimiter and stores each result in an array, the contents are trimmed after split. The array contains objects each having the properties “PartNumber", “PartText" and “PartLength".

Attributes
Into Specifies a name of the array-variable to store the result set. Defaults to “Array" when not specified. – Used in conjuction with action ForEachRow. **By** Specifies the text to use as delimiter to split the input text.

```
<SplitText By="<BR>" Into="Result">[[My_Text_Var]]</SplitText>
```

♦ Switch

Executes an action based on the value specified. This element contains one or more <Case Value=""> elements. A “Default" element can also be present.

Attributes

Value Value to test and to find an appropriate case.

```
<Switch Value="[[SOME_VAR]]">
     <Case Value="1">
        ...
     </Case>

     <Case Value="2">
        ...
     </Case>

     <Default>
        ...
     </Default>
</Switch>
```

♦ Subroutine

Defines a subroutine. Subroutines can be called later in code at any moment and as many times as desired.

Attributes
Name Name of the subroutine.

```
<Subroutine Name="MySub">
    Actions ...
</Subroutine>
```

♦ CallSubroutine

Calls a subroutine. The current context will be available to the subroutine. From within a subroutine executing <Stop/> will cause the subroutine to return.

Attributes
Name Name of the subroutine.

```
<CallSubroutine Name="MySub"/>
```

♦ Call

Calls another XML configuration file. Useful if you have processes in different files. The entire context will be shared with the specified config file.

```
<Call>MyFile.xml</Call>
```

♦ IFilterLoadText

Loads the plaint-text contents from any file using the IFilter interface. The respective IFilter must be installed on your machine. A new variable **FileText** will be introduced into the context with the plain-text contents of the file.

Attributes
Path Specifies the path to the file. If the file does not exist no error will be thrown, instead empty varibles will be introduced. (Unless Strict is set). **Strict** Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception will be thrown. Same will happen if an empty path is specified. Defaults to FALSE.

```
<IFilterLoadText Path="Info.doc"/>
```

♦ PdfLoadText

Loads the plaint-text contents from a PDF file using Poppler's pdftohtml utility and then parsing the HTML contents. A new variable **FileText** will be introduced into the context with the plain-text contents of the file.

Attributes
Path Specifies the path to the file. If the file does not exist no error will be thrown, instead empty varibles will be introduced. (Unless Strict is set). **Strict** Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception

will be thrown. Same will happen if an empty path is specified. Defaults to FALSE.

```
<PdfLoadText Path="Info.doc"/>
```

♦ SendMail

Sends an email. SMTP parameters are required. It also supports internal elements: <Subject>, <Message>, <To>, <Bcc>, <Cc>, and <Attachment> elements (the latter each with a filename to attach). A single <Message> element must be present with the message contents. – Note that the <To>, <Bcc>, <Cc> and <Attachment> elements can be present more than once, or they can have their contents separated by “\|” in a single-element.

Attributes
Host Host address of the SMTP server. **Port (optional)** Port of the SMTP server (defaults to 465). **SSL (optional)** Boolean value (TRUE/FALSE) indicating if SSL should be used (defauls to TRUE). **Username** Name of the user to use to login to the SMTP server. **Password** Password of the user to login to the SMTP server. **Silent (optional)** When set to TRUE, causes any error during the statement execution to be ignored and written to the log file instead of throwing the exception. Defaults to FALSE. **From** Email address of the sender. **FromName (optional)** Display name to be shown as sender.

To Email address of recipient. **Subject (optional)** Subject line.

```
<SendMail Host="" Username="" Password="" From="" To=""
Subject="">
    <Attachment>hello_world.dat</Attachment>
    <Message>
         <b>Hello World!</b>
    </Message>
</SendMail>
```

♦ PdfMerge

Merges one or more PDF files into a single PDF file.

Attributes
Output Specifies the path to the output file. The file will be created. **Input** Specifies a list of input PDF files to be merged. Each file must be separated by a | (ALT+124) symbol. If one of the files does not exist no error will be thrown (unless Strict is set) but the output file will not be created. **Strict** Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception will be thrown. Same will happen if an empty path is specified. Defaults to FALSE for silent failure.

```
<PdfMerge Output="X.pdf" Input="A.pdf|B.pdf|C.pdf"/>
```

♦ Sleep

Puts the process to sleep for a certain amount of time.

Attributes

Time Specifies the amount of milliseconds to sleep.

```
<Sleep Time="1000"/>
```

♦ **ReplaceText**

Replaces a string by another in the specified innerText.

Attributes
Old Value to search in the input string (to be replaced). **New** Value to use as replacement. **Into** Name of the output variable (defaults to Text).

```
<ReplaceText Into="Z" Old="X" New="Y">XYXYAZ</ReplaceText>
```

♦ **MsgLoadInfo**

Loads information from a MSG file in Outlook format. All information obtained including attachments will be available as variables.

Attributes
Path Specifies the path to the MSG file. If the file does not exist no error will be thrown, instead empty varibles will be introduced. (Unless Strict is set). **Strict** Boolean value (TRUE/FALSE) indicating if the file should be always verified to exist, when set to TRUE and the file does not exist an exception will be thrown. Same will happen if an empty path is specified. Defaults to FALSE. **Prefix** Specifies a prefix to be prepended on all introduced variables. When set all variables will have the name “X.Y" where “X" is the prefix and “Y" is the

variable name. Note that variables inside an array will not have this prefix applied (i.e. MSG_ATTACHMENTS).

Introduced Context Variables
MSG_SUBJECT Subject of the email. **MSG_BODY_TEXT** Text contents of the email's message body. **MSG_RCPT_TO** Email addresses found in the "To" header (separated by "|"). **MSG_RCPT_CC** Email addresses found in the "Cc" header (separated by "|"). **MSG_RCPT_BCC** Email addresses found in the "Bcc" header (separated by "|"). **MSG_RCPT** All the email addresses found in any of the email headers (To, Cc, or Bcc) separated by "|" character. **MSG_ATTACHMENTS** An array of attachments, can be traversed using ForEachRow, and each object in this array contains the following two fields: **MSG_ATTACHMENT_FILENAME** The file name of the attachment. **MSG_ATTACHMENT_DATA** Contents of the attachment. If the size (in bytes) of this ByteArray is required, the COUNT function can be used to determine it, such as: [[COUNT MSG_ATTACHMENT_DATA]]. ***NOTE:*** *The MSG_ATTACHMENT_DATA value will be provided as a ByteArray, to actually use it as a string you have to use [[STRING MSG_ATTACHMENT_DATA]], if you want a HexString you can use [[HEXSTR MSG_ATTACHMENT_DATA]].*

```
<MsgLoadText Path="Info.msg"/>
```