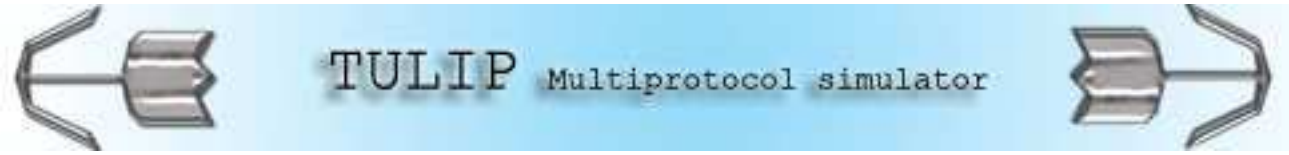


# TULIP command guide

(Baseline version 1.0)

## Table of content :

# MISCELLANEOUS (tlUtils.dll)

## 1. SEQ

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

Any command.

**Role :**

Body of the scenario (launcher, responder, traffic agent), encloses all the other commands. SEQ anchor should start every test sheet.

**Example :**

```
<SEQ>
     <PRINT>Hello World</PRINT>
</SEQ>
```

**Result:**

```
Hello World
```

## 2. EXEC

**Library :** tlUtils.dll

**Attributes :**

**FILE** : filename of the test sheet (plain text, usually ends with .xml), in global or relative format (if no directory, will assume the test sheet is located together with the caller test sheet.

**Content :**

None.

**Role :**

Execute a test sheet within the current execution scope (global variables). If the test sheet was compiled together with the tulip.xml file, it will also use the same constants.

**Example :**

```
<SEQ>
     <EXEC FILE="ABODR.XML"/>
</SEQ>
```

**Result:**

File ABODR.XML will be executed as if the commands were part of the caller test sheet. In case the file des not exists, there is a non blocking error message.

## 3. SPAWN_AGENT

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria.
**VALUE**: value of the selector used.

**Content :**

Any command.

**Role :**

Starts concurrents thread to the current one, each dedicated to a traffic agent. The attributes provide a flexible way to select which traffic agent to start.
The agents will execute in loop,depending on SCHEDULER parameters in tulip.xml. The fields selected are the agent attributes as defined in tulip.xml file.
From that time, all agent attributes can be accessed as normal variables.

**Example :**

```
<SEQ>
     <SPAWN_AGENT SELECTOR="TYPE" VALUE="CALLER">
          <PRINT>URI of caller traffic agent is ~URI~</PRINT>
     </SPAWN_AGENT>
</SEQ>
```

**Result:**

```
URI of caller traffic agent is test1
URI of caller traffic agent is test2
URI of caller traffic agent is test3
```

## 4. STATE

**Library :** tlUtils.dll

**Attributes :**

**NUM**: index of the current state.

**Content :**

None.

**Role :**

Works as a label for the current agent status. This is used for test result analysis, as it is kept in log each time an error occurs.
The state label is global to the agent in execution, each time it is set it will not change until set again or if the agent loops back. By default it is equal to 0.
The STATE is an agent attribute, it can be obtained anytime like any variable

**Example :**

```
<SEQ>
     <!-- INIT commands →
     <STATE NUM="1"/>
     <PRINT>Current state is ~STATE~</PRINT>
     <!-- Dialling commands →
     <STATE NUM="2"/>
     <PRINT>Current state is ~STATE~</PRINT>
</SEQ>
```

**Result:**

```
Current state is 1
Current state is 2
```

In case of error it will appear as F[x], where x is the agent state when the error happened.

## 5. DECLARE

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

VARIABLE anchors, one per local variable declaration.

**Role :**

Appears once at most in every test sheet, and allows local variable declaration. Each time the agent is restarted, or the test sheet invoked, these variables are back to their initialization value.

**Example :**

```
<SEQ>
  <DECLARE>
       <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
       <VARIABLE NAME="test" TYPE="STRING" VALUE=""/>
  </DECLARE>

  <PRINT>num_call = ~num_call~</PRINT>
</SEQ>
```

**Result:**

```
num_call = 0
```

## 6. VARIABLE

**Library :** tlUtils.dll

**Attributes :**

**NAME**: name of the variable to declare.

**TYPE**: enumerated type of the variable, can be one of the following.

*INTEGER*: plain text 32 bits integer, will be displayed in numeric format.

*STRING*,*STR*: plain text string, the length is not specified. Can be empty.

*FLOAT*: plain text 32 bits floating number. Is displayed as such.

*BOOLEAN, BOOL*: Boolean, can be true/1/TRUE or false/0/FALSE.

*BYTE*: ASCII one byte value, displayed in hexadecimal format.

*WORD*: ASCII two bytes value, displayed in hexadecimal format.

*DWORD*: ASCII four bytes value, displayed in hexadecimal format.

*ASCII*: ASCII text string, the length is not specified. Can be empty. Displayed in hexadecimal format.

**VALUE**: initialization value (can be changed afterwards).

**Content :**

None.

**Role :**

Used within DECLARE block, and represents a local variable declaration.

**Example :**

```
<SEQ>
  <DECLARE>
       <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
       <VARIABLE NAME="test" TYPE="STRING" VALUE=""/>
  </DECLARE>

  <PRINT>num_call = ~num_call~</PRINT>
</SEQ>
```

**Result:**

```
num_call = 0
```

## 7. RESPONDER

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

RECEIVE statements (one or more).

**Role :**

Used to enclose responder RECEIVE statements, and is enclosed only within SEQ anchor. There is at most one RESPONDER statement per scenario file.

**Example :**

```
<SEQ>
  <RESPONDER>
    <RECEIVE>
          <MESSAGE>5/**/</MESSAGE>
          <MATCH>
            <SEND>ACK</SEND>
          </MATCH>
    </RECEIVE>
  </RESPONDER>
</SEQ>
```

**Result:**

When a message starting with “5” (SIP error message) is sent to responder, the message ACK is sent as an answer.

## 8. RECEIVE

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

One MESSAGE statement + one MATCH statement.

**Role :**

Contains one action/answer block, and is enclosed within a RESPONDER anchor. The message represents the pattern to match: if so, the paired MATCH statement is executed, otherwise we move on to the next RECEIVE block.

**Example :**

```
<SEQ>
  <RESPONDER>
    <RECEIVE>
            <MESSAGE>5/**/</MESSAGE>
            <MATCH>
              <SEND>ACK</SEND>
            </MATCH>
    </RECEIVE>
  </RESPONDER>
</SEQ>
```

**Result:**

When a message starting with “5” (SIP error message) is sent to responder, the message ACK is sent as an answer.

## 9. MATCH

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

Any set of commands executable within a responder (not blocking).

**Role :**

Groups the commands executed when the pattern string is matched in the paired MESSAGE statement.

**Example :**

```
<SEQ>
  <RESPONDER>
    <RECEIVE>
          <MESSAGE>5/**/</MESSAGE>
          <MATCH>
            <SEND>ACK</SEND>
          </MATCH>
    </RECEIVE>
  </RESPONDER>
</SEQ>
```

**Result:**

When a message starting with “5” (SIP error message) is sent to responder, the message ACK is sent as an answer.

## 10. START_RESPONDER

**Library :** tlUtils.dll

**Attributes :**

**FILE** : filename of the responder test sheet (plain text, usually ends with .xml), in global or relative format (if no directory, will assume the test sheet is located together with the caller test sheet.

**Content :**

None.

**Role :**

Starts the responder execution, if it hasn't been started before, using the file name provided. The file will be executed each time the responder is invoked. There is one general purpose responder process + one dedicated responder per traffic agent, each has its own local variables scope.

**Example :**

```
<SEQ>
      <START_RESPONDER FILE="RESPONDER_SIP.xml"/>
</SEQ>
```

**Result:**

Message indication that the responder was started.

## 11. FAULT

**Library :** tlUtils.dll

**Attributes :**

**PACKET_VAR** : name of the variable which will contain the last received packet (usually, the one causing the error).

**Content :**

Any commands executed after fault detection.

**Role :**

Enclosed directly in SEQ statement (at most one per test sheet), it is invoked each time a blocking fault is triggered during traffic agent execution, before looping back.

**Example :**

```
<SEQ>
     <DECLARE>
            <VARIABLE NAME="fault_packet" TYPE="STRING" VALUE=""/>
     </DECLARE>
     <FAULT PACKET_VAR=”fault_packet” />
          <PRINT> --- Fault triggered at state ~STATE~ --- </PRINT>
          <PRINT> Received: ~fault_packet~ </PRINT>
     </FAULT>
</SEQ>
```

**Result:**

```
--- Fault triggered at state 0 ---
Received: (message)
```

## 12. SET_AGENT_PARAM

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**SELECTORVALUE**: value of the selector used. By default, "CURRENT"

**PARAM**: name of the agent parameter which will be added/modified.

**PARAM VALUE**: value of the parameter (string).

**Content :**

None.

**Role :**

Create/modify a string variable which will be maintained each time the agent restarts (traffic agent scope). By default, the test agent from which the function is executed will only be selected, unless it is executed from the launcher/commander, in which case all traffic agents will be modified (same result with SELECTOR=NO).

**Example :**

```
<SEQ>
   <SET_AGENT_PARAM PARAM="CALL_ID" PARAMVALUE="called_~NUM_AGENT~"/>
   <PRINT>CALL_ID=~CALL_ID~</PRINT>
</SEQ>
```

**Result:**

```
CALL_ID=called_1
```

## 13. SUSPEND

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**VALUE**: value of the selector used. By default, "CURRENT"

**Content :**

None.

**Role :**

Suspends the traffic agent selected, if it is running. By default, the test agent from which the function is executed will only be selected, unless it is executed from the launcher/commander, in which case all traffic agents will be suspended (same result with SELECTOR=NO).

**Example :**

```
<SEQ>
   <SUSPEND SELECTOR="NUM_AGENT" VALUE="2"/>
</SEQ>
```

**Result:**

```
Test agent 2 will be suspended.
```

## 14. RESUME

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**VALUE**: value of the selector used. By default, "CURRENT"

**Content :**

None.

**Role :**

Resume the traffic agent selected, if it is suspended. By default, the test agent from which the function is executed will only be selected, unless it is executed from the launcher/commander, in which case all traffic agents will be resumed if possible (same result with SELECTOR=NO).

**Example :**

```
<SEQ>
   <RESUME SELECTOR="NUM_AGENT" VALUE="2"/>
</SEQ>
```

**Result:**

Test agent 2 will be resumed.

## 15. STOP

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**VALUE**: value of the selector used. By default, "CURRENT"

**Content :**

None.

**Role :**

Cleanly stops the traffic agent selected: if it is running, waits for the end of the current call in order to make sure resources are properly released. By default, the test agent from which the function is executed will only be selected, unless it is executed from the launcher/commander, in which case all traffic agents will be stopped (same result with SELECTOR=NO).

**Example :**

```
<SEQ>
   <STOP SELECTOR="TYPE" VALUE="CALLER"/>
</SEQ>
```

**Result:**

The selected test agents will be stopped.

## 16. KILL

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**VALUE**: value of the selector used. By default, "CURRENT"

**Content :**

None.

**Role :**

Kills the traffic agent selected: if it is running, terminates it at the end of the current command. By default, the test agent from which the function is executed will only be selected, unless it is executed from the launcher/commander, in which case all traffic agents will be stopped (same result with SELECTOR=NO).

**Example :**

```
<SEQ>
   <KILL SELECTOR="TYPE" VALUE="CALLER"/>
```

```
</SEQ>
```

**Result:**

The selected test agents will be killed.

## 17. EXIT

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

Stops all running process agents, and cleanly exits the application.

**Example :**

```
<SEQ>
   <EXIT/>
</SEQ>
```

**Result:**

The application is closed.

## 18. SET_COLLECTOR

**Library :** tlUtils.dll

**Attributes :**

**ALL_AGENTS**: displays an entry for each traffic agent (default is "*CURRENT*"). It can be *YES* or *NO*.

**AGENT_SUMMARY**: displays the statistic summary line each time (total number of calls, faults,..). Can be *YES* or *NO*, default is "*CURRENT*".

**AGENT_DETAIL**: displays the detail for each test agent (fault counters broken down by status). Can be *YES* or *NO*, default is "*CURRENT*".

**LOG_ANALYSIS**: keeps a log of the statistics, in order to be able to perform log analysis (see command TRAFFIC_ANALYSIS). Can be *YES* or *NO*, default is "*CURRENT*".

**RESPONDER_LOG**: also show a statistic entry for the responder. Can be *YES* or *NO*, default is "*CURRENT*".

**PERIOD**: periodicity of statistic output. Should be either numeric (in seconds), or "*CURRENT*" (default).

**Content :**

None.

**Role :**

Configure dynamically the statistic collector.

**Example :**

```
<SEQ>
```

```
   <SET_COLLECTOR PERIOD="20"/>
</SEQ>
```

**Result:**

The change is visible at next statistic collection, which will now happen every 20 seconds.

## 19. SET_SCHEDULER

**Library :** tlUtils.dll

**Attributes :**

**INTERSIMU**: inter-simulation interval, time interval between two calls. Should be numeric, or "*CURRENT*" (by default).

**UNIT**: time unit for intersimulation value. Can be S (second) or MS (milliseconds) or "*CURRENT*" (by default)

**AGENT_DETAIL**: displays the detail for each test agent (fault counters broken down by status). Can be *YES* or *NO*, default is "*CURRENT*".

**SIMULTANEOUS**: number of calls which are started each time INTERSIMU*SIMULTANEOUS is reached, it is used to group call attempts and simulate a burst. Should be numeric, or "*CURRENT*" (by default).

**PERIODS**: maximum number of calls before the application is stopped.

**Content :**

None.

**Role :**

Configure dynamically the statistic collector.

**Example :**

```
<SEQ>
   <SET_SCHEDULER INTERSIMU="100" UNIT="MS"/>
</SEQ>
```

**Result:**

The change is visible at next statistic collection, which will show that call rate is now 10 CAPS.

## 20. TRACE

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "CURRENT"

**VALUE**: value of the selector used. By default, "CURRENT"

**LEVEL**: debug level applied to the agent, can be 0 to 4. By default, 2.

**Content :**

None.

**Role :**

Configure dynamically the debug level for selected agents. By default, the current test agent will be modified if command is executed from traffic agent, otherwise all of them will (same result if SELECTOR=NO).

**Example :**

```
<SEQ>
   <TRACE LEVEL="3"/>
</SEQ>
```

**Result:**

Debug level is set to 3 (very verbose) on all test agents.

## 21. DUMP_AGENT

**Library :** tlUtils.dll

**Attributes :**

**NUM_AGENT**: agent number ("*CURRENT*" by default).

**Content :**

None.

**Role :**

Display all parameters + general information about selected agents. By default, the current test agent will be displayed if command is executed from traffic agent, otherwise all of them will (same result if SELECTOR=NO).

**Example :**

```
<SEQ>
   <DUMP_AGENT NUM_AGENT="3"/>
</SEQ>
```

**Result:**

```
INFO  14:43:02 : (1) MGC AGENT
                 | TYPE = MGC
                 | ID = MGC-2
                 | NUM_AGENT = 2
                 | NB_AGENTS = 60
                 | STATE = 0
                 | CALLER_URI = +33296233931
                 | CALLER_NUM_AGENT = 22
                 | CALLED_URI = +33296234001
                 | CALLED_NUM_AGENT = 42
                 | MGC_NUM_AGENT = 2
                 | Current status : STATUS_WAITING
                 | NB_PERIODS=1
                 | NB_FAULTS=0   QF=1.000000
                 | | UPTIME=40.7s        EXEC TIME=0.4s
                 | AVG EXEC TIME=0.4s    AVG WAITING TIME=40.3s
```

## 22. TRAFFIC_ANALYSIS

**Library :** tlUtils.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

Performs a traffic analysis, provided the scheduler **LOG_ANALYSIS** field was set to true in tulip.xml .

**Example :**

```
<SEQ>
   <TRAFFIC_ANALYSIS/>
</SEQ>
```

**Result:**

```
STAT  15:29:01 : Analysis of traffic log (traffic time : 2413
seconds)
Total number of calls  :  241312
Total number of faults :  0
Base QF                :  1.000000
Average rate           :  50.002487 CAPS

Incident analysis:
Number of real faults computed:    0
Number of relevant calls computed: 241312
Computed QF:                       1.000000
```

## 23. GET_NB_PERIODS

**Library :** tlUtils.dll

**Attributes :**

**SELECTOR**: name of the test agent parameter used as the selection criteria. By default, "*CURRENT*"

**VALUE**: value of the selector used. By default, "*CURRENT*"

**VAR**: name of the variable where the number of periods will be stored.

**Content :**

None.

**Role :**

Outputs the number of calls into a variable. By default the current test agent is interrogated, otherwise the variable gets the sum of calls related to selected agents.

**Example :**

```
<SEQ>
   <GET_NB_PERIODS VAR="num_call"/>
   <PRINT> Number of periods is ~num_call~"/>
</SEQ>
```

**Result:**

```
Number of periods is 0
```

# Protocol specific (xxx.dll)

These commands are advertised by protocol specific libraries (ex, SIP.dll, MEGACO.dll,…). They do not have necessarily to be implemented within those libraries, in which case they would override the default declarations.

## 1. EXPECTED

**Library :** protocol specific library

**Attributes :**

**METHOD**: by default, "DATA". When applicable, application type of message expected (ex REQUEST, REPLY).

**TAM**: number of seconds during which the message will be expected, before firing an error.

**Content :**

The string to match, including input variables, joker, constants...

**Role :**

Executed within a traffic agent scenario: waits for the arrival of a message matching the enclosed string. If not, it is sent to the agent responder for a last chance check before firing an error. If no message arrives before timeout, an error is fired.

**Example :**

```
<SEQ>
     <EXPECTED TAM="5">
SIP/2.0 100 Trying
Call-ID: ~call_id~
To: "~CALLED_URI~" <sip:~CALLED_URI~@~HOME_DOMAIN~>
From: "~URI~" <sip:~URI~@~HOME_DOMAIN~>;tag=~from_tag~
CSeq: 1 INVITE
     </EXPECTED>
</SEQ>
```

**Result:**

The message needs to be received as specified, otherwise there will be an error.

## 2. SEND

**Library :** protocol specific library

**Attributes :**

**METHOD**: by default, "DATA". When applicable, application type of message sent (ex REQUEST, REPLY).

**Content :**

The string to send, including variables/constants.

**Role :**

Executed within a test agent scenario: sends a message. This is a non-blocking call.

**Example :**

```
<SEQ>
  <RESPONDER>
     <RECEIVE>
          <MESSAGE>
SIP/2.0 5~$$~
Via: SIP/2.0/UDP ~IP_TULIP~:5060;branch=~(branch)~
From: <sip:~(from)~@~(from_domain)~
To: <sip:~(to)~@~(to_domain)~
Call-ID: ~(CALL_ID)~
CSeq: ~(cseq)~ ~$$~

          </MESSAGE>
          <MATCH>
               <SEND>
ACK sip:~to~@~IP_MGC~:5060 SIP/2.0
Via: SIP/2.0/UDP ~IP_TULIP~:5060;branch=~branch~
From: <sip:~from~@~from_domain~
To: <sip:~to~@~to_domain~
Call-ID: ~CALL_ID~
CSeq: ~cseq~ ACK
Content-Length: 0

               </SEND>
          </MATCH>
     </RECEIVE>
  </RESPONDER>
</SEQ>
```

**Result:**

In this case, SEND is used within a responder test sheet. If a SIP 5xx error message is received, the corresponding ACK is sent.

### 3. CHANGE_PORT

**Library :** protocol specific library

**Attributes :**

**PORT**: new port value.

**Content :**

None.

**Role :**

Dynamically change the protocol port when applicable.

**Example :**

```
<SEQ>
  <CHANGE_PORT PORT="2945">
</SEQ>
```

**Result:**

In this case, the subsequent messages will be sent to remote port 2945.

### 4. TEST_SEND

**Library :** protocol specific library

**Attributes :**

**PIPELINE**: destination of test message, can be "*INPUT*" or "*COMMAND*". Default is "INPUT", which simulates the incoming media. "*COMMAND*" represents the Tulip commands pipeline.

**Content :**

The string to send, including variables/constants.

**Role :**

Executed within a test agent scenario: sends a message. This is a non-blocking call.

**Example :**

```
<SEQ>
     <TEST_SEND>
ACK sip:~to~@~IP_MGC~:5060 SIP/2.0
Via: SIP/2.0/UDP ~IP_TULIP~:5060;branch=~branch~
From: <sip:~from~@~from_domain~
To: <sip:~to~@~to_domain~
Call-ID: ~CALL_ID~
CSeq: ~cseq~ ACK
Content-Length: 0

     </TEST_SEND>
```

```
</SEQ>
```

**Result:**

In this case, TEST_SEND is used within a test agent scenario, to simulate the reception of ACK SIP message.

### 5. TEST_RECEIVE

**Library :** protocol specific library

**Attributes :**

**METHOD**: by default, "DATA". When applicable, application type of message expected in the test pipeline (ex REQUEST, REPLY).

**TAM**: number of seconds during which the message will be expected, before firing an error.

**Content :**

The string to match, including input variables, joker, constants...

**Role :**

Executed within a test agent scenario: waits for the arrival of a message matching the enclosed string from the test pipeline. If no message arrives before timeout, an error is fired.

**Example :**

```
<SEQ>
     <TEST_RECEIVE TAM="5">
SIP/2.0 100 Trying
Call-ID: ~call_id~
To: "~CALLED_URI~" <sip:~CALLED_URI~@~HOME_DOMAIN~>
From: "~URI~" <sip:~URI~@~HOME_DOMAIN~>;tag=~from_tag~
CSeq: 1 INVITE
     </TEST_RECEIVE>
</SEQ>
```

**Result:**

The message needs to be received as specified, otherwise there will be an error.

## 6. MESSAGE

**Library :** protocol specific library

**Attributes :**

None.

**Content :**

A pattern string.

**Role :**

Contains the string (including input variables, joker) which will be matched against the packet sent to responder.

**Example :**

```
<SEQ>
  <RESPONDER>
    <RECEIVE>
          <MESSAGE>5/**/</MESSAGE>
          <MATCH>
            <SEND>ACK</SEND>
          </MATCH>
    </RECEIVE>
  </RESPONDER>
</SEQ>
```

**Result:**

When a message starting with “5” (SIP error message) is sent to responder, the message ACK is sent as an answer.

# COMMON (tlCommon.dll)

These commands are used for general shared functions (arithmetic, logic, ..) which can be used by any test case.

## 1. SWITCH

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of variable used in the switch comparison.

**Content :**

None.

**Role :**

Perform a decision based on the content of a variable, to decide which enclosed “CASE” block to choose from, if none complies then choose ”DEFAULT”.

**Example :**

```
<SEQ>
  <SET VAR=”error_code” VALUE=”1”/>
  <SWITCH VAR=”error_code”>
    <CASE VALUE=”1”>
          <PRINT>Error 1 received</PRINT>
    </CASE>
    <CASE VALUE=”3”>
          <PRINT>Error 3 received</PRINT>
    </CASE>
    <DEFAULT>
          <PRINT>Default error handling</PRINT>
    </DEFAULT>
  </SWITCH>
</SEQ>
```

**Result:**

```
Error 1 received
```

## 2. CASE

**Library :** tlCommon.dll

**Attributes :**

**VALUE**: value of variable used in the switch comparison.

**Content :**

The set of commands executed in case of match..

**Role :**

If the variable in the SWITCH statement equals this value, then the commands enclosed will be executed.

**Example :**

```
<SEQ>
  <SET VAR="error_code" VALUE="1"/>
  <SWITCH VAR="error_code">
    <CASE VALUE="1">
          <PRINT>Error 1 received</PRINT>
    </CASE>
    <CASE VALUE="3">
          <PRINT>Error 3 received</PRINT>
    </CASE>
    <DEFAULT>
          <PRINT>Default error handling</PRINT>
    </DEFAULT>
  </SWITCH>
</SEQ>
```

**Result:**

```
Error 1 received
```

## 3. DEFAULT

**Library :** tlCommon.dll

**Attributes :**

None.

**Content :**

The set of commands executed in case default is selected..

**Role :**

If the variable in the SWITCH statement cannot match any CASE values, then the commands enclosed will be executed.

**Example :**

```
<SEQ>
  <SET VAR="error_code" VALUE="7"/>
  <SWITCH VAR="error_code">
    <CASE VALUE="1">
           <PRINT>Error 1 received</PRINT>
    </CASE>
    <CASE VALUE="3">
           <PRINT>Error 3 received</PRINT>
    </CASE>
    <DEFAULT>
           <PRINT>Default error handling</PRINT>
    </DEFAULT>
  </SWITCH>
</SEQ>
```

**Result:**

```
Default error handling
```

### 4. SET

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will be set.
**VALUE**: new value of variable, including variables/constants.

**Content :**

None.

**Role :**

Sets the content of a variable. In case the value is not consistent with the variable type, a non-blocking error will be raised.

**Example :**

```
<SEQ>
  <SET VAR="error_code" VALUE="1"/>
  <PRINT>Error ~error_code~ received</PRINT>
</SEQ>
```

**Result:**

```
Error 1 received
```

### 5. CAT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will be set.
**VALUE**: value to be added to the variable, including variables/constants.

**Content :**

None.

**Role :**

If the variable in the SWITCH statement equals this value, then the commands enclosed will be executed.

**Example :**

```
<SEQ>
  <SET VAR="test" VALUE="This is a"/>
  <CAT VAR="test" VALUE=" new test"/>
  <PRINT_VAR VAR="test"/>
</SEQ>
```

**Result:**

```
This is a new test
```

### 6. MIDDLE

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will contain the result.
**MIN**: minimum index to be selected (inclusive).
**MAX**: maximum index to be selected (exclusive).

**Content :**

The string where the MIDDLE will to applied to.

**Role :**

The variable in the attribute will receive the extract of the content string between the minimum index (starts with 0) and the maximum one.

**Example :**

```
<SEQ>
  <MIDDLE VAR="test" MIN="2" MAX="6">This is a test</MIDDLE>
  <PRINT_VAR VAR="test"/>
</SEQ>
```

**Result:**

```
is i
```

### 7. LEFT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will contain the result.
**MIN**: minimum index to be selected (inclusive).

**Content :**

The string where the LEFT will to applied to.

**Role :**

The variable in the attribute will receive the extract of the content string after the minimum index (starts with 0).

**Example :**

```
<SEQ>
  <LEFT VAR="test" MIN="2">This is a test</MIDDLE>
  <PRINT_VAR VAR="test"/>
</SEQ>
```

**Result:**

```
is is a test
```

## 8. RIGHT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will contain the result.
**MAX**: maximum index to be selected (exclusive).

**Content :**

The string where the RIGHT will to applied to.

**Role :**

The variable in the attribute will receive the extract of the content string before the maximum index (starts with 0).

**Example :**

```
<SEQ>
  <MIDDLE VAR="test" MAX="6">This is a test</MIDDLE>
  <PRINT_VAR VAR="test"/>
</SEQ>
```

**Result:**

```
This i
```

## 9. IF

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will be tested.
**VALUE**: value to compare the variable with.
**OPER**: one of the allowed operands.
*LESS*: true if the variable is less than the value.
*MORE*: true if the variable is greater than the value.
*EQUAL*: true if the variable is equal to the value.
*NEQUAL*: true if the variable is not equal to the value.
*STARTS, START*: true if the variable starts with the value.
*ENDS, END*: true if the variable ends with the value.
*CONTAINS, CONTAIN*: true if the variable contains the value.

**Content :**

Mandatory THEN statement, optional ELSE statement.

**Role :**

Perform the comparison of a variable with a value, using a specified operand. In case the comparison is successful the THEN block is executed, otherwise the ELSE will be executed if present.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="14"/>
  </DECLARE>

  <IF VAR="i" OPER="EQUAL" VALUE="14">
    <THEN>
      <PRINT>i = 14  !!! then block </PRINT>
    </THEN>
    <ELSE>
      <PRINT>i <> 14  !!! else block </PRINT>
    </ELSE>
  </IF>
</SEQ>
```

**Result:**

```
i = 14  !!! then block
```

## 10. THEN

**Library :** tlCommon.dll

**Attributes :**

None.

**Content :**

Enclosed commands.

**Role :**

Is included within IF block. Encloses the command which will be executed if the IF statement comparison is successful.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="14"/>
  </DECLARE>

  <IF VAR="i" OPER="EQUAL" VALUE="14">
    <THEN>
      <PRINT>i = 14  !!! then block </PRINT>
    </THEN>
    <ELSE>
      <PRINT>i <> 14  !!! else block </PRINT>
    </ELSE>
  </IF>
</SEQ>
```

**Result:**

```
i = 14  !!! then block
```

## 11. FOR

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will be used in the loop.
**MIN**: starting value of the variable.
**MAX**: ending value of the variable, has to be greater than MIN.

**Content :**

Enclosed commands.

**Role :**

Loops the enclosed commands (MIN-MAX) times. Each time, the variable is incremented.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <FOR VAR="i" MIN="1" MAX="5">
    <PRINT> Value of i : ~i~</PRINT>
  </FOR>
</SEQ>
```

**Result:**

```
Value of i : 1
Value of i : 2
Value of i : 3
Value of i : 4
```

## 12. WHILE

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable which will be tested.
**VALUE**: value to compare the variable with.
**OPER**: one of the allowed operands.
*LESS*: true if the variable is less than the value.
*MORE*: true if the variable is greater than the value.
*EQUAL*: true if the variable is equal to the value.
*NEQUAL*: true if the variable is not equal to the value.
*STARTS, START*: true if the variable starts with the value.
*ENDS, END*: true if the variable ends with the value.
*CONTAINS, CONTAIN*: true if the variable contains the value.

**Content :**

Enclosed commands.

**Role :**

Perform the comparison of a variable with a value, using a specified operand. While the comparison is successful the enclosed commands are looped.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <WHILE VAR="i" OPER="LESS" VALUE="20">
    <ADD VAR="i" VALUE="2"/>
    <PRINT_VAR VAR="i"/>
  </WHILE>
</SEQ>
```

**Result:**

```
15
17
19
```

## 13. ADD

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable modified.
**VALUE**: value added to the variable.

**Content :**

None.

**Role :**

Add a specified value to the variable.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <ADD VAR="i" VALUE="2"/>
  <PRINT_VAR VAR="i"/>
</SEQ>
```

**Result:**

```
17
```

## 14. SUBTRACT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable modified.
**VALUE**: value subtracted to the variable.

**Content :**

None.

**Role :**

Subtract a specified value to the variable.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <SUBTRACT VAR="i" VALUE="2"/>
  <PRINT_VAR VAR="i"/>
</SEQ>
```

**Result:**

```
13
```

## 15. MULTIPLY

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable modified.
**VALUE**: value subtracted to the variable.

**Content :**

None.

**Role :**

Subtract a specified value to the variable.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <MULTIPLY VAR="i" VALUE="2"/>
  <PRINT_VAR VAR="i"/>
</SEQ>
```

**Result:**

```
30
```

## 16. DIVIDE

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable modified.
**VALUE**: value subtracted to the variable.

**Content :**

None.

**Role :**

Subtract a specified value to the variable.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <MULTIPLY VAR="i" VALUE="3"/>
  <PRINT_VAR VAR="i"/>
</SEQ>
```

**Result:**

```
5
```

### 17. PRINT_VAR

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be printed.

**Content :**

None.

**Role :**

Outputs a variable content (nothing else).

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="i" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>

  <PRINT_VAR VAR="i"/>
</SEQ>
```

**Result:**

```
VARIABLE i=15
```

### 18. PRINT

**Library :** tlCommon.dll

**Attributes :**

None.

**Content :**

The string to output (including variables/constants).

**Role :**

Outputs a string.

**Example :**

```
<SEQ>
  <PRINT>Hello WORLD</PRINT>
</SEQ>
```

**Result:**

```
Hello WORLD
```

## 19. TEMPO

**Library :** tlCommon.dll

**Attributes :**

**NB**: number of time units.
**UNIT**: time unit, either MS (milliseconds, default) or S (seconds).

**Content :**

None.

**Role :**

Put the current agent in sleep mode for a specified duration.

**Example :**

```
<SEQ>
  <PRINT>Hello WORLD</PRINT>
  <TEMPO UNIT="S" NB="10"/>
  <PRINT>Hello again WORLD</PRINT>
</SEQ>
```

**Result:**

```
Hello WORLD
Hello again WORLD
```

The string "Hello again WORLD" takes 10s to appear.

## 20. LENGTH

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**Content :**

The string to evaluate, including variables/constants.

**Role :**

Store the length of the enclosed string and store it into a specified variable.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>
  <LENGTH VAR=”length”>Hello WORLD</LENGTH>
  <PRINT>Length is ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is 11
```

## 21. CHAR_AT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**INDEX**: index of the character to be extracted.

**Content :**

The string to evaluate, including variables/constants.

**Role :**

Extract one character at a specified position within the enclosed string, and store it into the variable..

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="char" TYPE="STRING" VALUE="15"/>
  </DECLARE>
  <CHAR_AT VAR=”char” INDEX=”2”>Hello WORLD</LENGTH>
```

```
  <PRINT>Extracted ~char~</PRINT>
</SEQ>
```

**Result:**

```
Extracted l
```

### 22. CLEARSCREEN

**Library :** tlCommon.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

When applicable, clears the output screen.

**Example :**

```
<SEQ>
  <CLEARSCREEN/>
</SEQ>
```

**Result:**

The screen is cleared.

### 23. AND

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.
**VALUE**: value to be applied to the variable.

**Content :**

None.

**Role :**

Modify the variable using the AND operator. The variable should be either integer or hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="15"/>
  </DECLARE>
```

```
    <AND VAR="length" VALUE="8"/>
    <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 8
```

### 24. NOT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**Content :**

None.

**Role :**

Modify the variable using the NOT operator. The variable should be either integer or hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="BYTE" VALUE="15"/>
  </DECLARE>
  <NOT VAR=”length” />
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 240
```

### 25. OR

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**VALUE**: value to be applied to the variable.

**Content :**

None.

**Role :**

Modify the variable using the OR operator. The variable should be either integer or hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="4"/>
  </DECLARE>
  <OR VAR=”length” VALUE=”1”/>
  <PRINT>Length is now ~length~</PRINT>
```

```
</SEQ>
```

**Result:**

```
Length is now 5
```

## 26. XOR

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.
**VALUE**: value to be applied to the variable.

**Content :**

None.

**Role :**

Modify the variable using the XOR operator. The variable should be either integer or hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="BYTE" VALUE="4F4F"/>
  </DECLARE>
  <XOR VAR=”length” VALUE=”123A”/>
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 5D 75
```

## 27. MODULO

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.
**MOD**: value which will be applied with the MODULO.
**VALUE**: value to be applied to the variable.

**Content :**

None.

**Role :**

Perform a modulo operation of *VALUE* with *MOD*. The variable should be either integer or hexadecimal. The result is stored in the variable specified.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="25"/>
  </DECLARE>
  <MODULO VAR=”length” MOD=”4” VALUE=”~length~”/>
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 1
```

## 28. SHIFT

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.
**DIRECTION**: direction of the bit shift (either *LEFT* or *RIGHT*).
**NB**: number of bits.

**Content :**

None.

**Role :**

Perform a bit shifting operation (direction and number of bits specified). The variable should be either integer or hexadecimal. The result is stored in the variable specified.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="25"/>
  </DECLARE>
  <SHIFT VAR=”length” DIRECTION=”RIGHT” NB=”3”/>
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 24
```

## 29. CHANGE_ENDIAN

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**Content :**

None.

**Role :**

Perform a bit rotation operation (change of endian). The variable should be either integer or hexadecimal. The result is stored in the variable specified.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="INTEGER" VALUE="128"/>
  </DECLARE>
  <CHANGE_ENDIAN VAR=”length” />
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 8
```

## 30. TO_HEXA

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**Content :**

None.

**Role :**

Set a variable to a value interpreted as hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="DWORD" VALUE="01020304"/>
  </DECLARE>
  <TO_HEXA VAR=”length”/>148AB167</TO_HEXA>
  <PRINT>Length is now ~length~</PRINT>
```

```
</SEQ>
```

**Result:**

```
Length is now 14 8A B1 67
```

## 31. TO_HEXA

**Library :** tlCommon.dll

**Attributes :**

**VAR**: name of the variable to be modified.

**Content :**

None.

**Role :**

Set a variable to a value interpreted as hexadecimal.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="length" TYPE="DWORD" VALUE="01020304"/>
  </DECLARE>
  <TO_HEXA VAR="length"/>148AB167</TO_HEXA>
  <PRINT>Length is now ~length~</PRINT>
</SEQ>
```

**Result:**

```
Length is now 14 8A B1 67
```

# System (tlSystem.dll)

## 1. GET_SYSTEM

**Library :** tlSystem.dll

**Attributes :**

**VAR**: name of variable.

**PARAM**: name of system parameter, can be one of the following.

*NB_FAULTS* : total number of faults
*NB_PERIODS* : total periods number
*NB_AGENTS_CFG* : number of agents declared
*NB_AGENTS_RUN* : number of agents started
*TIME* : the time of day (milliseconds)
*DATE* : the date DD:MM:YYYY
*UPTIME* : number of seconds since the computer was started
*UPTIME_TULIP* : number of seconds since tulip was started
*HOSTNAME* : the computer host name
*LOCAL_IP* : the local IP(s) used for traffic
*CPU_CAPA* : CPU capacity in MHz
*CPU_USED* : CPU percentage used by all appli
*CPU_TULIP* : CPU usage for this tulip instance
*MEM_CAPA* : memory capacity in Mbytes
*MEM_USED* : memory used by all appli
*MEM_TULIP* : memory usage for the tulip instance

**Content :**

None.

**Role :**

Interrogate some system-related values, or global tulip parameters.

**Example :**

```
<SEQ>
  <DECLARE>
     <VARIABLE NAME="nb_periods" TYPE="INTEGER" VALUE="0"/>
  </DECLARE>
  <GET_SYSTEM PARAM=”NB_PERIODS” VAR=”nb_periods”/>
  <PRINT>~nb_periods~ calls performed/>
</SEQ>
```

**Result:**

```
0 calls performed
```

# Time (tlTime.dll)

## 2. START_WATCH

**Library :** tlTime.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

Starts a watch dedicated to the agent executing the function. The subsequent ON_WATCH will use this as a time reference.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
     </DECLARE>

     <GET_NB_PERIODS VAR="num_call"/>
     <IF VAR="num_call" OPER="EQUAL" VALUE="0">
          <THEN>
               <START_WATCH/>
               <PRINT>Initialize WATCH</PRINT>
          </THEN>
     </IF>
     <ON_WATCH NB="10" UNIT="S">
          <PRINT>10s watch ! resetting watch ...</PRINT>
     </ON_WATCH>
</SEQ>
```

**Result:**

```
Initialize WATCH
10s watch ! resetting watch ...
```

The second line appears 10s later.

## 3. STOP_WATCH

**Library :** tlTime.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

Destroy a running watch dedicated to the agent executing the function. It is not usable anymore until recreated.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
     </DECLARE>

     <GET_NB_PERIODS VAR="num_call"/>
     <IF VAR="num_call" OPER="EQUAL" VALUE="0">
          <THEN>
               <START_WATCH/>
               <PRINT>Initialize WATCH</PRINT>
          </THEN>
     </IF>
     <ON_WATCH NB="10" UNIT="S">
          <PRINT>10s watch ! stop watch ...</PRINT>
          <STOP_WATCH/>
     </ON_WATCH>
</SEQ>
```

**Result:**

```
Initialize WATCH
10s watch ! resetting watch ...
```

The second line appears 10s later, the watch cannot be stopped or observed again.

## 4. RESET_WATCH

**Library :** tlTime.dll

**Attributes :**

None.

**Content :**

None.

**Role :**

Resets a running watch dedicated to the agent executing the function. The time reference is set to the current time.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
     </DECLARE>

     <GET_NB_PERIODS VAR="num_call"/>
     <IF VAR="num_call" OPER="EQUAL" VALUE="0">
          <THEN>
               <START_WATCH/>
               <PRINT>Initialize WATCH</PRINT>
          </THEN>
     </IF>
     <ON_WATCH NB="10" UNIT="S">
          <PRINT>10s watch ! stop watch ...</PRINT>
          <RESET_WATCH/>
     </ON_WATCH>
</SEQ>
```

**Result:**

```
Initialize WATCH
10s watch ! resetting watch ...
10s watch ! resetting watch ...
10s watch ! resetting watch ...
...
```

The second line loops every 10s.

## 5. ON_WATCH

**Library :** tlTime.dll

**Attributes :**

**NB**: number of time units.
**UNIT**: time unit, either MS (milliseconds, default) or S (seconds).

**Content :**

Any enclosed command.

**Role :**

Checks the elapsed time on the agent watch (if it was started before) and compare it with a specified value. In case it exceeds that value, the enclosed commands are executed.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="num_call" TYPE="INTEGER" VALUE="0"/>
     </DECLARE>

     <GET_NB_PERIODS VAR="num_call"/>
     <IF VAR="num_call" OPER="EQUAL" VALUE="0">
          <THEN>
               <START_WATCH/>
               <PRINT>Initialize WATCH</PRINT>
          </THEN>
     </IF>
     <ON_WATCH NB="10" UNIT="S">
          <PRINT>10s watch ! stop watch ...</PRINT>
          <RESET_WATCH/>
     </ON_WATCH>
</SEQ>
```

**Result:**

```
Initialize WATCH
10s watch ! resetting watch ...
10s watch ! resetting watch ...
10s watch ! resetting watch ...
…
```

The second line loops every 10s.

# Crypto (tlCrypto.dll)

## 1. MD5SUM

**Library :** tlCrypto.dll

**Attributes :**

**VAR**: name of variable.

**Content :**

The string which will be checksumed, including variables/constants.

**Role :**

Computes the MD5 checksum of the string enclosed, and store the result in the variable specified.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="checksum" TYPE="STRING" VALUE=""/>
     </DECLARE>

     <MD5SUM VAR=”checksum”>coucou</MD5SUM>
     <PRINT>MD5 Checksum of “coucou” is : ~checksum~</PRINT>
</SEQ>
```

**Result:**

```
MD5 Checksum of “coucou” is : 721a9b52bfceacc503c056e3b9b93cfa
```

## 2. SHA1SUM

**Library :** tlCrypto.dll

**Attributes :**

**VAR**: name of variable.

**Content :**

The string which will be checksumed, including variables/constants.

**Role :**

Computes the SHA-1 checksum of the string enclosed, and store the result in the variable specified.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="checksum" TYPE="STRING" VALUE=""/>
     </DECLARE>

     <SHA1SUM VAR=”checksum”>coucou</MD5SUM>
     <PRINT>SHA1 Checksum of “coucou” is : ~checksum~</PRINT>
</SEQ>
```

**Result:**

```
SHA1 Checksum of “coucou” is :
5ed25af7b1ed23fb00122e13d7f74c4d8262acd8
```

## 3. RANDOM

**Library :** tlCrypto.dll

**Attributes :**

**VAR**: name of variable.
**FORMAT**: format of the random number, either *INTEGER* or *HEXA*, default is *INTEGER*.
**LENGTH**: length of the random number, default is unspecified ("0").

**Content :**

None.

**Role :**

Generates a random number based on the CPU timestamp + agent number. It is possible to choose between a HEXA or INTEGER format, and the length of output, which is stored in the variable provided.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="nonce" TYPE="STRING" VALUE=""/>
     </DECLARE>

     <RANDOM VAR=”nonce” LENGTH=”10”/>
     <PRINT>Generated nonce: ~nonce~</PRINT>
</SEQ>
```

**Result:**

```
Generated nonce: 5ed25af7b1
```

## 4. TO_BASE64

**Library :** tlCrypto.dll

**Attributes :**

**VAR**: name of variable.

**Content :**

The string which will be converted, including variables/constants.

**Role :**

Computes the BASE64 encoding of the string enclosed, and store the result in the variable specified.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="base64" TYPE="STRING" VALUE=""/>
     </DECLARE>

     <TO_BASE64 VAR=”base64”>coucou</MD5SUM>
     <PRINT>BASE64 result is: ~base64~</PRINT>
</SEQ>
```

**Result:**

```
BASE64 result is: Y291Y291
```

## 5. CRC32SUM

**Library :** tlCrypto.dll

**Attributes :**

**VAR**: name of variable.

**Content :**

The string which will be checksumed, including variables/constants.

**Role :**

Computes the CRC32 checksum of the string enclosed, and store the result in the variable specified.

**Example :**

```
<SEQ>
     <DECLARE>
          <VARIABLE NAME="checksum" TYPE="STRING" VALUE=""/>
     </DECLARE>

     <CRC32SUM VAR=" checksum">coucou</MD5SUM>
     <PRINT>CRC32 Checksum of "coucou" is: ~checksum~</PRINT>
</SEQ>
```

**Result:**

```
CRC32 Checksum of "coucou" is: f0baebc7
```