2018

WATS STANDARD XML FORMAT MANUAL

DESCRIBES THE WSXF FORMAT AND HOW TO USE IT

VIRINCO AS

CONTENTS

GENERAL DETAILS

WATS format name	Wats Standard XML Format (WSXF)
Version	1.1
File example name	WatsStandardXMLFormat.xml
Last modified date	03.10.2018

This format is a standardised format that the WATS Client will automatically read and import into WATS (was first release in 2014).

The WATS Standard XML Format follows the WSXF XML schema.

Each report consists of two main parts; a **Header data** part, a **Step data** part. The Header data part is further split into two sub parts; a list of predefined headers followed by an optional section of miscellaneous UUT data.

The WSXF Converter receives an Xml file with UUT or UUR data. It goes through the elements in the file and creates a UUT/UUR report in the TDM API with all the tests and measurements included. The WATS Client can read one or multiple UUT report(s) per file.

XML

WSFX is XML format. XML is made up of elements. An XML element has an opening tag with a name and attributes, a closing tag, and content.

The opening tag of an element starts with a < character followed by the name of the tag. After the name comes an optional list of attributes for the element. Attributes are defined with the name of the attribute, then the value of the attribute in quotes after an equals sign. The opening tag ends with a > character.

```
<Report type="UUT" Start="2018-01-01T08:30:00" Result="Passed">
```

The closing tag of an element starts with </ followed by the name of the element, and ends with a > character.
WSXF uses <Report> element to organize the data.

```
</Report>
```

Some elements have no content and can be self-closing. In this case the opening tag ends with />

```
<Process Code="10" />
```

Between the opening and closing tags of an element is the contents of the element. The content can be text or other elements. Elements inside other elements are called nested elements and are children of the element they are nested in. All child elements must be closed before the parent can be closed.

```
<Reports>
   <Report type="UUT" Start="2018-01-01T08:30:00" Result="Passed"> ←Child of reports
      <Process Code=”10” />                                          ← Child of report 1
   </Report>
   <Report type="UUT" Start="2018-01-01T08:40:00" Result="Passed"> ← Child of reports
      <Process Code=”10” />                                          ← Child of report 2
   </Report>
</Reports>
```

CONVERTER METHOD

The WATS API supports two methods for importing data: Active and Import. When operating in Active mode, the result of a step, sequence and test is set automatically using the measure, compare operator and the limit(s). In Import mode, the status (pass/fail) must be set manually.

The Wats Standard XML Format converter use a combination of Active and Import. If status is given in the data; it will be used. If not, limits are used. See below for rules and recommendation regarding this.

NUMERIC AND DATE FORMATS

Numeric: By default . (period) is used as decimal separator, so PI should be written as “3.14159”. Do not use thousand separators. Only digits and decimal point is allowed.

DateTime:

Use the ISO 8601 format (2018-09-12T14:26:16.977+02:00) e.g.

If requested, support for custom numeric / date formats can be added.

REPORTS

A WSXF file should start with a <Reports> element. <Reports> can contain one or more <Report> elements, each containing data from one report.

```
<Reports
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://wats.virinco.com/schemas/WATS/Report">
</Reports>
```

HEADER DATA

The Header data part of the file consists of two parts. The first part is three elements containing basic information, such as *SerialNumber* and *PartNumber,* about the UUT. The second part is any miscellaneous data elements that contains additional information such as sub-reports and misc info linked to the UUT/UUR Report.

Required information (Header)

```
<Report
      type="UUT"
      Start="1900-01-01T01:00:00.000+01:00"
      SN="FATTest-268" PN="FATPartNo"
      Rev="Rev1" >
   <UUT UserLoginName=”FAToper” />
   <Process Code="60" Name="FST" />
```

REPORT ELEMENT

Below is a list of the <Report>element attributes. It is recommended to fill in as much data as possible as this significantly improves the usability of the final WATS report and analysis of report data. Note that either “Start” or “Start_UTC” is required.

Attribute Name	Description	Data Format	Importance
Type	The type of report.	String (UUT or UUR)	Required
SN (Serial Number)	Serial number of the unit.	String(100)	Required
PN (Part Number)	Part number of the unit.	String(100)	Required
Start	Start time in local time. If missing, calculated based on Start_utc.	String(Formated according to ISO 8601): 2018-09-12T14:26:16.977+02:00	Required (can fill in «Start_utc» instead.
Result	Report result	See Status Codes	Required
Rev	Revision of the unit.	String(100)	Recommended
Start_utc	Start time in UTC time zone. If missing, calculated based on Start.	String (See Start)	Optional
Location	Location where the test takes place.	String(100)	Optional
Machine Name	Name of test station. If missing, the computer name will be used.	String(100)	Optional
ID	A Globally Unique ID of the report. A report submitted with the same ID as another will overwrite the report. If missing will be generated.	GUID	Optional
Origin	Origin of the report. Automated test equipment identifier	GUID	Optional
Purpose	Automated test equipment purpose (Report origin)	String(100)	Optional

IMPLEMENTATION EXAMPLE

```
<Report type="UUT" Start="1900-01-01T01:00:00.000+01:00" Start_utc="1900-01-01T00:00:00.000Z"
        Result="Failed" SN="FATTest-268" PN="FATPartNo" Rev="Rev1" MachineName="FAT Station"
        Location="FAT Station Location" Purpose="FAT Station Purpose">
```

UUT ELEMENT

If the report is a UUT report, the <Report> element may contain a <UUT> element. The UUT element contains additional header information regarding the UUT report. It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report.

Attribute Name	Description	Data Format	Importance
UserLoginName	Name of the operator of the test system.	String(100)	Recommended
Execution Time	Total duration of test in seconds. If missing, will be generated from execution time of each step.	Double(100)	Recommended
BatchSN	Batch serial number of the unit.	String(100)	Optional
TestSocketIndex	When parallel or batch execution.	Short	Optional
FixtureId	ID of the fixture used to perform the tests.	String(100)	Optional
ErrorCode	Error code of any error that occurred during the test, if any.	Int(32)	Optional
ExecutionTime	Time from the execution started untill it finished	Double	Optional
ErrorMessage	Message of the error that occurred during the test, if any.	String(500)	Optional

The UUT may contain a <Comment> Element.

Element Name	Description	Data Format	Importance
Comment	Basic text comment that can be attached to the UUT report header	String(5000)	Optional

IMPLEMENTATION EXAMPLE

```
<UUT UserLoginName="FAToper" BatchSN="BatchSN" TestSocketIndex="999"
    ExecutionTime="123.67" FixtureId="FAT FixtureId" ErrorCode="999"
    ErrorMessage="Error code 999">
  <Comment>This is a Comment</Comment>
</UUT>
```

PROCESS ELEMENT

The <Report> element must contain a <Process> element. The process defined the test operation for the UUT report, and Repair operation for UUR reports. You can create and manage Processes in Control panel -> Configure/Settings -> Processes. Note that the checkbox for “Is test operation” must be checked for UUT report processes.

Attribute Name	Description	Data Format	Importance
Code	Operation type code for a WATS process.	Short	Required / Optional if Name is supplied
Name	Name of the operation type for a WATS process.	String(500)	Optional / required if Code is not supplied

IMPLEMENTATION EXAMPLE

```
<Process Code="10" Name="SW Debug" />
```

MISCINFO ELEMENT

Any further data concerning the UUT or the general test run must be entered as miscellaneous UUT data. It can for example be used for software/firmware versions, or for whatever purpose suits the test. There can be any number of <MiscInfo> elements in a <Report>. String data should the content of the element.

Attribute Name	Description	Data Format	Importance
Description	Name of the miscellaneous info	String(100)	Required
Typedef	Defition of the miscinfo type.	String(30)	Optional
Numeric	If the value of the info is numeric it should be set here.	Short	Optional

Data	Description	Data Format	Importance
Miscinfo content	Miscellaneous element content	String(100)	Optional

IMPLEMENTATION EXAMPLE

```
<MiscInfo Typedef="" Description="MiscInfoNumeric" Numeric="100"/>
<MiscInfo Typedef="" Description="MiscInfoString>Hello</MiscInfo>
<MiscInfo Typedef="" Description="InvalidXML" >This is a test with &lt; and &gt;</MiscInfo>
```

SUB UNITS / REPORTUNITHIERARCHY

You can attach sub-units to the test report, which will be linked in the report header. This allow you track systems and their individual component, which gives traceability to the systems lifecycle.

Register a sub unit by adding a <ReportUnitHierarchy> element to a <Report> with these attributes.

Attribute name	Description	Data (Format)	Importance
PN	Part number	String(100)	Required
SN	Serial number	String(100)	Required
Rev	Revision	String(100)	Optional
PartType	Part type	String(50)	Optional

IMPLEMENTATION EXAMPLE

```
<ReportUnitHierarchy PartType="PCBA" PN="SubPartNo1" SN="SubPartSN1"
                     Rev="Rev2" />
```

STEP DATA

The second part of the XML file contains the actual test step data. Data is entered in nested step tags following the XML Syntax. The first step in the XML file is the root step and must therefore only contain a sequence call. The table below describes the attributes which is required/optional for a <Step>.

Attribute Name	Description	Data Format	Importance
Group	Specifies the step group	Setup, Main, Cleanup	Required
Name	Name of the test step.	String(100)	Required & Unique
Status	The outcome of the test step	See Status Codes	Required
total_time	Duration of the step in seconds.	Double	Recommended
Start	Step start /date/time	DateTime (String)	Optional
StepCaused UUTFailure	Set to 1 if step caused the final test status to fail.	0, 1	Optional
StepCausedSequence Failure	Indicates if this step caused the parent sequence to fail.	Boolean	Optional
StepType	Step type or category, see the different step types for value	Defined in each step type below	Optional
StepErrorMessage	Message from external system with details regarding the error	String(200)	Optional
StepErrorCode	Integer error code	Int(32)	Optional

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="ActionStep" total_time="0" StepCausedUUTFailure="0"
Status="Passed" StepType="Action" />
```

STEP TYPES

Step type	Description	Dailed information
SequenceCall	Used to create Hiearachy structure	SequenceCall
NumericLimit	Numeric limit test	NumericLimit
StringValue	String value test	StringValue
PassFail	Pass/Fail test	PassFail
Chart	Graphs with data series	Chart
CallExe	Logs data from a call .exe step	CallExe
Attachment	Attachments such as README, Photoes etc	Attachment
MessagePopup	Logs data from a message popup step	MessagePopup
Action	Used to log test action data and information	Action

SEQUENCE STEPS

Sequence steps are steps which contains a sequence call. Sequence calls are used to create hierarchy structured data. A sequence step can have the following attributes:

Attribute Name	Description	Data Format	Importance
Version	Version of the sequence file	String(30)	Recommend
Filename	Filename of the sequence file	String(200)	Recommend
Filepath	Path to the sequence file	String(500)	Recommend
Name	Name of the sequence. If left blank, it will be set to Partnumber from Report header.	String(500)	Optional

Implementation example for Sequence calls

```
<Step Group="Main" Name="Pass/Fail tests" StepType="SequenceCall"
Status="Failed">
   <SequenceCall Name="Pass/Fail tests" Version="1.0.0" Filename="name"/>
   <Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
   <Step Group="Main" Name="SinglePass1" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
   <Step Group="Main" Name="SinglePass2" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
</Step>
```

Pass/Fail tests

- SinglePass
- SinglePass1
- SinglePass2

Sequence calls can also be nested, to support even more detailed tests.

```
<Step Group="Main" Name="Pass/Fail tests" StepType="SequenceCall" Status="Failed">
   <SequenceCall Name="Pass/Fail tests" Version="1.0.0" Filename="name"/>
   <Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
   <Step Group="Main" Name="SinglePass1" StepType="ET_PFT" Status="Passed">
     <SequenceCall Name="Nested Pass/Fail" Version="1.0.0" Filename="name"/>
     <Step Group="Main" Name="SinglePass2.1" StepType="ET_PFT" Status="Passed">
       <PassFail Status="Passed" />
     </Step>
     <Step Group="Main" Name="SinglePass2.2" StepType="ET_PFT" Status="Passed">
       <PassFail Status="Passed" />
     </Step>
   </Step>
</Step>
```

Pass/Fail tests

- Test1.1
- Nested Pass/Fail
  - Test2.1
  - Test2.2

TEST STEPS

A step containing a measurement is defined as a test step. There are three test step types:

- NumericLimitTest (Keyword: ET_NLT)
  Used to log numeric test data. the required fields are numericValue, units and CompOperator. Name must be present for multiple tests, but not for a single test.

- StringValueTest (Keyword: ET_SVT)
  Used to log string value test data. The required fields are CompOperator, String Value and Status. Name must be present for multiple tests, but not for a single test.

- PassFailTest (Keyword: ET_PFT)
  Used to log pass/fail test data. The required fields are status field. Name must be present for multiple tests, but not for single test. Step types

A step with one of these types must contain the corresponding measurement element.

Test steps comes in two different forms:

1. Single steps (one measurement) (*recommended*).
   Name attribute is optional
2. Multiple steps (two or more measurements):
   Name attribute **MUST** be set and have a unique name within the step.

Test steps must have status attribute filled in (Passed/Failed). For single steps it is also possible to use Skipped, Done, Terminated and Error.

This table explains the proper keyword to use for single and multiple steps.

Type	Keyword (single)	Keyword (multiple)
MultipleNumericLimitTest	StepType="ET_NLT"	StepType="ET_MNLT"
MultipleStringValueTest	StepType="ET_SVT"	StepType="ET_MSVT"
MultiplePassFailTest	StepType="ET_PFT"	StepType="ET_MPFT"

MULTIPLE STEPS

A multiple step is when it is necessary to bundle together two or more measurements. This can be done using Single Step Syntax and include one of the following keywords in the “StepType” attribute. If the attribute is set to “measure” WATS will set the correct attribute based on the multiple step data.

The simplest way to write a multiple step, is to use the same syntax as for a single step and include a name for the given step type. Name must be set to different values for the different measurements in the multiple measurement step.

The main step status will be calculated based on the combined status of the sub-tests. If one of them fail, the step fails. Statuses ‘Done’, ‘Terminated’ and ‘Error’ will for both for the individual measurements and for the main step status be translated to failed. ‘Skipped’ will show for the individual measurement, but the main step will have Passed even if all measurements are Skipped.

IMPLEMENTATION EXAMPLE

```
<Step “StepType=”ET_MNLT” Status=”Failed”……>
 <NumericLimit Name="T1" NumericValue="1" CompOperator="LOG" Status="Passed"/>
 <NumericLimit Name="T1" NumericValue="2" CompOperator="LOG" Status="Failed"/>
</Step>

<Step “StepType=”ET_MNLT” Status=”Passed”……>
 <NumericLimit Name="T3" NumericValue="1" CompOperator="LOG" Status="Passed"/>
 <NumericLimit Name="T4" NumericValue="2" CompOperator="LOG" Status="Passed"/>
</Step>
```

1. In the first example above, Status will be calculated for both lines (to Passed and Failed). The main step status will be set based on these (to Failed).

2. In the second example above, Status will NOT be calculated for the lines (both set to Passed). The main step status will be set based on these (to Passed).

NUMERIC LIMIT

Numeric limit tests are used to test numeric values. Numeric limit tests are defined with a value and a lower or/and higher limit. WSXF also contains a Comparison operator to specify which type of tests it should run. Below is a table of NumericLimit attributes, and their importance.

Attribute Name	Description	Data Format	Importance
NumericValue	The measured value.	Numeric	Required
CompOperator	Which comparison operation to use on the measures value.	See Comparison Operations	Required
Status	If the test passed or failed.	See Status Codes	Required
Name	Name of the limit test.	String(100)	Optional / Required & Unique if multiple tests in same step.
LowLimit	Limit used to compare the measured value in greater than and equal/not equal comparisons.	Numeric	Optional / Required if comparison operation is GE, EQ, or NE
HighLimit	The high limit used to compare measured value in less than comparisons.	Numeric	Optional / Required if Compare Operator is LE
Units	Unit of the measured value.	String(20)	Optional

IMPLEMENTATION EXAMPLE

Single NumericLimit Test

```
<Step Group="Main" Name="SingleEQPass" StepType="ET_NLT" Status="Passed" total_time="0">
    <NumericLimit NumericValue="500" LowLimit="500" Units="V" CompOperator="EQ"
Status="Passed" />  </Step>
```

Multiple NumericLimit Test

```
<Step Group="Main" Name="Multiple" StepType="ET_MNLT"  total_time="0" StepCausedUUTFailure="0">
        <NumericLimit Name="MultiLog" NumericValue="10" Units="A"
                        CompOperator="LOG"  />
        <NumericLimit Name="MultiEQPass" NumericValue="500" LowLimit="500" Units="V"
                        CompOperator="EQ"  />
        <NumericLimit Name="MultiEQFail" NumericValue="500" LowLimit="501" Units="V"
                        CompOperator="EQ"  />
        <NumericLimit Name="MultiGEPass" NumericValue="500" LowLimit="500" Units="V"
                        CompOperator="GE"  />
</Step>
```

STRING VALUE

StringValue tests are used to compare strings. StringValue tests usually exists of a StringValue, StringLimit and a Comparison operator. Below is a table of StringValue attributes, and their importance.

Attribute Name	Description	Data Format	Importance
StringValue	The string to compare.	String(100)	Required
CompOperator	Which comparison operation to use on the measures value.	See Comparison Operations	Required
Status	If the test passed or failed.	See Status Codes	Optional
Name	Name of the stringValue test.	String(100)	Optional / Required & Unique if multiple tests in same step.
StringLimit	The string to compare against.	String(100)	Required / Optional if Comparison Operator is Logging

IMPLEMENTATION EXAMPLE

Single StringValue Test

```
<Step Group="Main" Name="SingleInsiensPass" StepType="ET_SVT" Status="Passed"  total_time="0">
    <StringValue CompOperator="IGNORECASE" StringLimit="CaseSensitive"
        StringValue="caseSensitive" Status="Passed" />
</Step>
```

Multiple StringValue Test

```
<Step Group="Main" Name="Multiple" StepType="ET_MSVT"  total_time="0" StepCausedUUTFailure="0">
      <StringValue Name="MultiLog" CompOperator="LOG" StringValue="Test123"  />
      <StringValue Name="MultiIgnore" CompOperator="IGNORECASE" StringLimit="CaseSensitive"
                  StringValue="casesensitive" />
      <StringValue Name="MultiCaseSenit" CompOperator="CASESENSIT" StringLimit="CaseSensitive"
                  StringValue="CaseSensitive"/>
</Step>
```

PASS FAIL

Attribute Name	Description	Data Format	Importance
Status	If the test passed or failed.	See Status Codes	Required
Name	Name of the pass-fail test.	String(100)	Optional / Required if multiple tests in same step.

IMPLEMENTATION EXAMPLE

Single PassFail Test

```
<Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed" total time="0">
        <PassFail Status="Passed" />
 </Step>
```

Multiple PassFail Test

```
<Step Group="Main" Name="Multiple" StepType="ET_MPFT"  total_time="0" StepCausedUUTFailure="0">
        <PassFail Name="Pass" Status="Passed"  />
        <PassFail Name="Fail" Status="Passed" />
        <PassFail Name="PassSkipped" Status="Passed"  />
</Step>
```

ACTION STEPS

Used to log test action data and information. These steps are always simple steps, never multiple. Steps with stepType Action may contain <ReportText> elements. These steps do not contain MeasureName, Value, LowLimit, HighLimit, CompOperator or Units. There can only be one <ReportText> element in each step.

Element Name	Description	Data Format	Importance
ReportText	Exdented text field	String(5000)	Optional

IMPLEMENTATION EXAMPLE

```
<Step Group="Setup" Name="ActionStep" StepType="NI_Notification" Status="Passed">
      <ReportText>NI Package Manager loaded successfully</ReportText>
</Step>
```

CHART

The WSXF converter supports chart/graph data steps. Charts can be added to blank steps, or as sub-steps of test steps. Chart consist of two elements: The chart frame, and the chart series (data).

CHART FRAME

The <Chart> element has the following attributes:

Attribute name	Description	Data (Format)	Importance
ChartType	Type of chart. Line is default.	Line, LineLogXY, LineLogX, LineLogY	Recommended
Label	Name of the chart	String(100)	Recommended
YUnit	Name of the X axis	String(20)	Optional
YLabel	Unit of the X axis	String(50)	Optional
XUnit	Name of the Y axis	String(20)	Optional
XLabel	Unit of the Y axis	String(50)	Optional

CHART SERIES

A Series element (or XY-plot) contains a semicolon separated list of x-values and y-values. X-values can be omitted for import and will in such case be generated string with 1 and incremented by 1. This version only supports the XYG datatype. Each series must atleast contain one <YData> element. This element must contain a semicolon-separated list of values to be plotted. If <XData> is empty, the values in <YData> will be plotted at their index on the X axis. Each chart can't have more than 10 series.

- YData and XData can't have more than 10000 entries each.
- The number of elements in Y and X must match.

Attribute	Description	Data (Format)	Importance
YData	All data for y-axis.	Semicolon-separated string without limits	Required
Name	Name of plot.	String(100)	Required
XData	All data for x-axis.	Semicolon-separated string without limits	Recommended. If empty, the x-axis will be added with incrementing elements (1,2,3,4 etc).
DataType	Attribute (only XYG supported)	String(10)	Optional.

IMPLEMENTATION EXAMPLE

As step type (Graph Step)

```
<Step Group="Main" Name="Fibionacci" StepType="WATS_XYGMNLT" Status="Passed">
      <Chart ChartType="Line" idx ="0" Label="Fibonacci" XLabel="X" XUnit="" YLabel="Y" YUnit="">
            <Series Name="Fibionacci" DataType="XYG">
               <ydata>0;1;1;2;3;5;8;13;21;34;55;89;144</ydata>
               <xdata>0;1;2;3;4;5;6;7;8;9;10;11;12</xdata>
            </Series>
            <Series Name="Mean" DataType="XYG">
               <ydata>28.923076;28.923076</ydata>
               <xdata>0;12</xdata>
             </Series>
      </Chart>
</Step>
```

Sub-step of test (NumericLimit test)

```
<Step Group="Main" Name="Fibionacci" StepType="WATS_XYGMNLT" Status="Passed">
    <NumericLimit NumericValue="28.923076" Units="Number" CompOperator="LOG"
                        Status="Passed" />
      <Chart ChartType="Line" idx ="0" Label="Fibonacci" XLabel="X" XUnit="" YLabel="Y" YUnit="">
            <Series Name="Fibionacci" DataType="XYG">
               <ydata>0;1;1;2;3;5;8;13;21;34;55;89;144</ydata>
               <xdata>0;1;2;3;4;5;6;7;8;9;10;11;12</xdata>
            </Series>
            <Series Name="Mean" DataType="XYG">
               <ydata>28.923076;28.923076</ydata>
               <xdata>0;12</xdata>
             </Series>
      </Chart>
</Step>
```

ATTACHMENT

The WSXF supports attachments. Attachments can be included in the WSXF file by using the <Attachment> keyword. WSXF uses Base64 encoded contents to attach a file. for handling attachments, and therefore require the attachment to be in the form of a byte-array.

Attribute	Description	Data (Format)	Importance
Name	Name for the attachment	String(100)	Required
ContentType	Type of attachment (mime-type)	String(100)	Required.
SizeField	Can be used to indicate if data output has been suppressed.	Int(32)	Optional.

The byte array is parsed as an element, not as an attribute.

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="ByteArray" StepType="WATSFile" Status="Passed">
        <Attachment Name="WiFiNotification" ContentType="image/png"
        >AAAAAAAAAAAElFTkSuQmCC</Attachment>
</Step>
```

MESSAGE POP UP

Message popup is used when a pop-up message is displayed.

Attribute	Description	Data (Format)	Importance
Button	Code for which button was pressed	Short	Optional
Response	Message from pop up	String(200)	Optional

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="Message Popup" StepType="MessagePopup" Status="Done" total_time="0">
      <MessagePopup Button="1" Response="Response text" />
</Step>
```

CALL EXECUTABLE

Step which contains a reference to an executable.

Attribute	Description	Data (Format)	Importance
ExitCode	ExitCode from the executable	Double	Required

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="CallExecutable" StepType="CallExecutable" Status="Done"
        total_time="0">
   <Callexe ExitCode="0" />
</Step>
```

COMPARISON OPERATORS

The table below describes the different Comparison operators for test types.

CompOperator	Description	Step type
EQ	Equal	Numeric, StringValue
NE	Not Equal	Numeric, StringValue
GT	Greater than	Numeric, StringValue
LT	Less than	Numeric, StringValue
GE	Greater or equal	Numeric, StringValue
LE	Less or equal	Numeric, StringValue
GTLT	Greater than low limit, less than high limit	Numeric
GELE	Greater or equal than low limit, less or equal than high limit	Numeric
GELT	Greater or equal than low limit, less than high limit	Numeric
GTLE	Greater than low limit, less or equal than high limit	Numeric
LTGT	Less than low limit, greater than high limit	Numeric
LEGE	Less or equal than low limit, greater or equal than high limit	Numeric
LEGT	Less or equal than low limit, greater than high limit	Numeric
LTGE	Less than low limit, greater or equal than high limit	Numeric
LOG	Loging values (no comparison)	Numeric, StringValue
CASESENSIT	Case sensitive	StringValue
IGNORECASE	Ignore case / !CASESENSIT	StringValue

STEP STATUS CODES

Below is a list of supported status codes for a step.

Status	UUT description	Step / measurement description	Additional information	Color code
Passed	UUT passed	Step passed		Green
Failed	UUT failed	Step failed		Red
Done	Not used.	Step completed.	Can be used for single steps, should not be used for multiple steps	Cyan
Skipped	Not used.	Step execution skipped	Can be used for single steps, with caution for multiple steps.	Yellow
Error	An error occurred during test execution	An error occurred during step execution	Can be used for single steps, should not be used for multiple steps.	Orange
Terminated	Operator terminated test execution	Operator terminated step execution	Can be used for single steps, should not be used for multiple steps.	Blue

The UUT report itself can be Passed, Failed, Error or Terminated.

A single step can have all statuses.

A single measurement (in a multiple step) can only be Passed or Failed or Skipped (but all Skipped measurements will end with Passed for main step). Done or Error or Terminated will be translated to Failed.

LOG FILE

Two folders are set up in the FilePath that is referred to in the Converters.xml file. These are:

- Done
- Error

When a file with the correct file type (.xml in this case) is added to the FilePath, it is picked up by the WATS Standard XML Converter. After the system has imported the data in the file, it is moved to one of the following folders:

- Done (if the process went ok)
- Error (if it did not)

There are two log-files found within Programdata\Virinco folder.

- wats.log; a file for the client which has data about the files that is picked up by the Service and imported through the converter.
- .error: a file that includes errors that occurred in the import process. The report will not be submitted in this case, and the input xml-file will br added to the Error-folder. The warning file will have the xml-file name before dot.
    - Example: Input file named SampleData.xml, the warning file will be named SampleData.error.

WSXF FILE EXAMPLE

A WSXF example file can be found here:

https://virinco.zendesk.com/hc/en-us/article_attachments/360010417831/WATS_-_WSXF_Example_file.xml