Fabasoft app.test Console Player

The Fabasoft app.test Console Player offers the possibility to start the execution of a test from the command line. Everything that is required is a simple configuration file.

Structure of the configuration file

```
<Config projectpath="D:\app.test Projects\Folio\">
    <Run            ... />
    <Login          ... />
    <Reporting      ... />
    <Telemetry      ... />
</Config>
```

Test run information

```
<Run
    Specify the test that should be executed
    execute="{~projectpath~}Tests\Folio.app-test"
    Specify the name of the server to connect to
    webserver="fscserver"
    Specify the default URL that should be used
    address="http://{~webserver~}/fsc"
    Specify the client that should be used
    (HTML_IE, HTML_MOZILLA : 3.0)
    clienttype="HTML_MOZILLA : 3.0"
    Specify the scope
    scope="2"
    Specify how errors should be handled
    onerror="break"
    Specify how breakpoints should be handled
    onbreakpoint="continue"
/>
```

Login information

```
<Login
    Specify the authentication mode that should be used
    (basic or integrated)
    authentication="basic"
    Specify the test user login
    username="anthony.brown"
    Specify the password of the test user
    password="anthony"
/>
```

Report information

```
<Reporting
    Specify the name of the report that should be created
    reportname="{~testname~}-{~scope~}.app-rep"
    Specify details about the client environment
    envclient="Microsoft Windows Vista"
/>
```

Fabasoft app.test

www.apptest.com

Application Testing Redefined

CHEAT SHEET

Execution

The `<Execution />` statement is used to perform various actions. This can be a click, double click, context menu, an import or export of a file, selections and so on.

The general syntax for execution is:

```
<Execution action="..." location="..." value="..."
    eval="..." options="..." if="..." />
```

This should be the first step in a Fabasoft Folio test

```
action="Click" location="PORTALS.Classic"
```

Select an object from the object list

```
action="Click" location="Object List[0]"
action="Click" location='Object List["Name"==
    "My Folder"]'
```

Open the context menu

```
action="Contextmenu" location="Created by"
```

Set the value of a property

```
action="Set" location="First Name"
value="Anthony"
```

Perform a quick search

```
action="Quicksearch" location="Organization"
value="Fabasoft*"
```

Import and export files

```
action="Import" value="{~importdocs~}
    Import.docx"
action="Click" value="{~exportdocs~}
Export.docx" options="Save"
```

Select objects in an object list

```
action="Click" location="Object List[0]"
action="Click" location="Object List[LAST]"
options="ShiftKey" or
action="Click" location="Object List[LAST]"
    options="CtrlKey"
```

Set

The `<Set />` statement is used to store values to temporary variables. These values can be accessed by using `{~parameter_name~}`.

The general syntax for execution is:

```
<Set parameter="..." location="..." value="..."
    eval="..." if="..." expression="..." />
```

Set a parameter to a static value

```
parameter="Organization" value="My Value"
```

Set a parameter based on the value of a control

```
location="Object List[LAST]"
```

Set true or false based on the existence of an element with label 'Next'

```
location="Next" eval="Exists"
```

Count the number of elements in the object list

```
location="Object List" eval="Count"
```

Store the COO address of an object from the object list

```
location="Object List[1]" eval="Address"
```

Set true or false depending whether the combo box "Sex" contains "Male" or not

```
location="Sex" eval='Contains("Male")'
```

Set parameters by processing strings

```
eval='SplitLeft("Anthony.Brown",".")'
eval='Substr("Fabasoft app.test",9,4)'
eval='Substr("Anthony Brown",
                " Brown ","Black")'
```

Set a parameter by resolving a Fabasoft Components expression

```
expression="coort.getObject({~COOAddress~}).
    COOSYSTEM@1.1:objectcreatedby" />
```

Validation

By enhancing tests through `<Validation />` statements you can make sure that assertions are met and the solution meets the expectations.

The syntax for validation statements is:

```
<Validation ok="..." if="..." />
```

Continue if two strings are equal

```
ok='"{~FirstName~} {~LastName~}"=="Anthony
    Brown"'
ok='"{~Substring~}"!="app.test"'
```

Continue if calculating and comparing two dates returns true

```
ok="{~datenow~}+365>=01.01.2014"
```

Continue if calculating and comparing two integers returns true

```
ok="{~Objects~}>0"
ok="{~Objects~}-1==0"
```

Continue if a Boolean value returns true or false

```
ok="{~StringComparison~}"
ok="!{~ListLength~}"
```

Validation returns true if Length is false and Substring equals app.test

```
ok='!{~Length~}&&"{~Substring~}"=="app.test"'
```

Validation returns true if StringComparison is true or Objects equals zero

```
ok="{~StringComparison~}||{~Objects~}==0"
```

Validation returns true if StringComparison is true or Object equals zero and the current date is not 01.02.2014

```
ok="{~StringComparison~}||{~Objects~}>0&&
    {~datenow~}!=01.02.2014"
```

Validation with a location attribute:

```
<Validation location="HEADER.PORTALS.Kontakte"
eval="exists"/>
```

```
<Validation location="HEADER.PORTALS"
eval="count" value="4"/>
<Validation ok="{~VALIDATION~} > 3"
location="HEADER.PORTALS" eval="count"/>
```

Conditions

By using conditions you can specify under which conditions statements should be executed. If the condition is not met, the statement is reported as "Not executed".

Execute the action if the element exists on the page

```
action="Click" location="Next" if="Exists"
```

Execute the step if the current date is 01.01.2014

```
if="{~datenow~}==01.01.2014"
```

Execute the step if the value of Organization is not Fabasoft

```
if='"{~Organization~}"!="Fabasoft"'
```

Execute the step if the value of the parameter ControlExists is false

```
if="!{~ControlExists~}"
```

Execute the step if the value of ControlExists is false and the current date is 01.01.2014

```
if="!{~ControlExists~}&&
    {~datenow~}==01.01.2014" />
```

Handle errors

You can specify if errors are expected. If the expected error occurs, the execution is OK.

Continue the test only if any error occurred

```
ok="Error"
```

Continue only if the error with the defined error message occurred

```
ok='Error("Property \"Surname\" must be
    defined")'
```

Continue no matter if an error occurred or not

```
ok="Always"
```