User Reference

The Conditional Command

Date: 1/2/2018

Revision: 3.5

Introduction

When executing a series of TestSteps, it is sometimes necessary to alter the sequence based on, for example the success or failure of a step. TAZ handles this with the `Conditional{…}` command. This command is similar to an if/else statement in traditional programming languages. `Conditional{…}` checks if one or more conditions are met, then takes the appropriate action. The syntax for the command is:

`Conditional{ArgA|ArgB|ArgC}`

Where ArgA is the condition, ArgB is the action to take if the condition **is** met, and ArgC is the action to take if the condition **is not** met.

The Conditions (ArgA)

The Condition in ArgA is either met or not met. ArgA may contain more than 1 condition, using the operators AND (&) or OR (^). You may combine Conditions in ArgA but you may **not** use both operators at the same time.

Here are a few of the more common Conditions to use in ArgA. See the TAZ Command Manual for the complete list.

Fail- This Condition checks the preceding `Evaluation` commands in current TestStep, beginning at the most recent command. If a failure **is** found, the condition is met and the flow proceeds to ArgB**.** If a failure **is not** found, the condition is not met and the flow proceeds to ArgC**.** The Fail argument may also take an additional parameter to specify which `Evaluation` commands to check, such as an integer list. `Evaluation` commands within the TestStep are numbered, beginning with 1 for the most recent command. If this parameter is omitted, any failure within the test step satisfies the condition.

In **Figure-1** The `Fail` condition is **met**. Two of the evaluation commands were successful, however the specified `Evaluation` command (#2) failed, and therefore the `Fail` condition is met. **ArgB** will determine the next steps.

```
Step1=
InsertText{cat dog bird}
Will produce the Return "cat dog bird"
-------------------------------------
Xpect{cat} - Third Checked (#3)
Xpect{fox} - Second Checked (#2)
Xpect{bird} - First checked (#1)
Conditional{Fail((2))|…} - This condition is
met
```

Figure 1

RSLT - This argument searches the **Return** for a specified string. The `RSLT` argument is followed by a parameter that contains the text to match.

In **Figure-2** The condition `RSLT((bird))` is met, because the string "bird" is included in the return

```
Step1=
InsertText{cat dog bird}
Will produce the Return "cat dog bird"
-------------------------------------
Xpect{fish}
Conditional{RSLT((bird))|…} - This condition
is met
```

Figure 2

Actions

After the Condition is evaluated `Conditional{…}` determines how the TestCase will proceed using Action arguments. ArgB specifies the action taken if the Condition **is** met. ArgC specifies the action to take if the Condition **is not** met.

Some of the possibilities for ArgB and ArgC are outlined below.

Skip- This action will skip the specified TestSteps. It has many different parameters which are used to specify which steps to skip. An integer, a comma delimited list, or range of integers, will directly specify the TestStep numbers to skip. In addition to integer arguments you can use the parameters **All** or **None**.

The action **Run** might be considered the opposite of **Skip**, but there is a very important difference. **Skip** starts with the premise that all remaining TestSteps will run and **Skip** indicates which one won't. Whereas, **Run** starts with the premise that all remaining TestSteps will not execute and **Run** indicates which ones will. This means that if the `Conditional` action is Run((6-7)) **only** TestSteps 6 and 7 will execute, regardless how many other TestSteps are in the TestCase.

Note that the `SubCase{…}` command will modify the step numbers in your TestCase. See `Conditional{…}` ArgB in the TAZ Command Manual for options to use with `SubCase{…}`.

In **Figure-3** the Pass condition is met. According to ArgB TAZ will skip `steps 2` and `3`, but will run all other TestSteps.

```
Step1=
InsertText{cat dog bird}
Will produce the Return "cat dog bird"
-------------------------------------
Xpect{cat}
Xpect{dog}
Xpect{bird}
Conditional{Pass|Skip((2,3))|…} - The
condition is met
```

Figure-3

NextStep – When the Condition is met, the action **NextStep** will cause the flow to immediately proceed to the next TestStep. It will skip all remaining commands in the current TestStep. When used in the `Loop{..}` command, this option can be used to break out of the loop when Conditions are met.

In **Figure-4** the condition is met, and the test will proceed to the next step in the TestCase, skipping all other commands. `InsertText{Fish}` will not be executed

```
Step1=
InsertText{cat dog bird}
Will produce the Return "cat dog bird"
-------------------------------------
Xpect{cat}
Xpect{dog}
Conditional{Pass|NextStep|…} - This condition
is met, The rest of the step is skipped
InsertText{Fish} - Produces the Return "Fish"
```

Figure-4

ContinueStep – This action will continue in the current TestStep if the appropriate condition is met. It is important to note that if the condition is not met, TAZ will skip all remaining commands in the TestStep. When used in the `Loop{..}` command, this option can be used to break out of the loop when Conditions are not met.

In **Figure-5** the pass condition is met and the test will proceed as expected to the `InsertText{mouse}` command.

```
Step1=
InsertText{cat dog bird}
Will produce the Return "cat dog bird"
-------------------------------------
Xpect{cat}
Conditional{Pass|ContinueStep|…} -This
condition is met, the rest of the TestStep is
executed
InsertText{mouse}
```

Figure-5

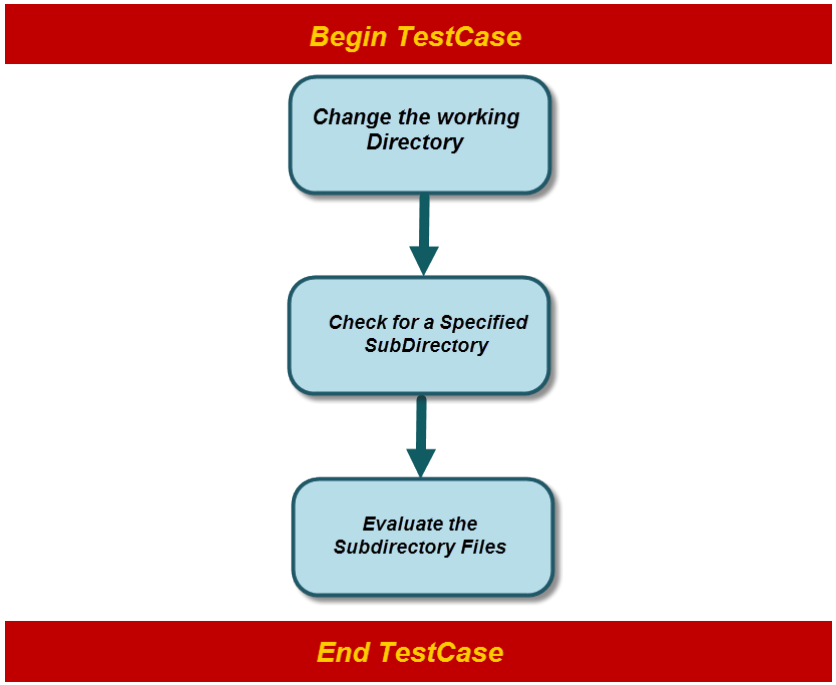
Implementing the Conditional Command

In this section we will demonstrate the implementation of the conditional command by building a **TestCase**.

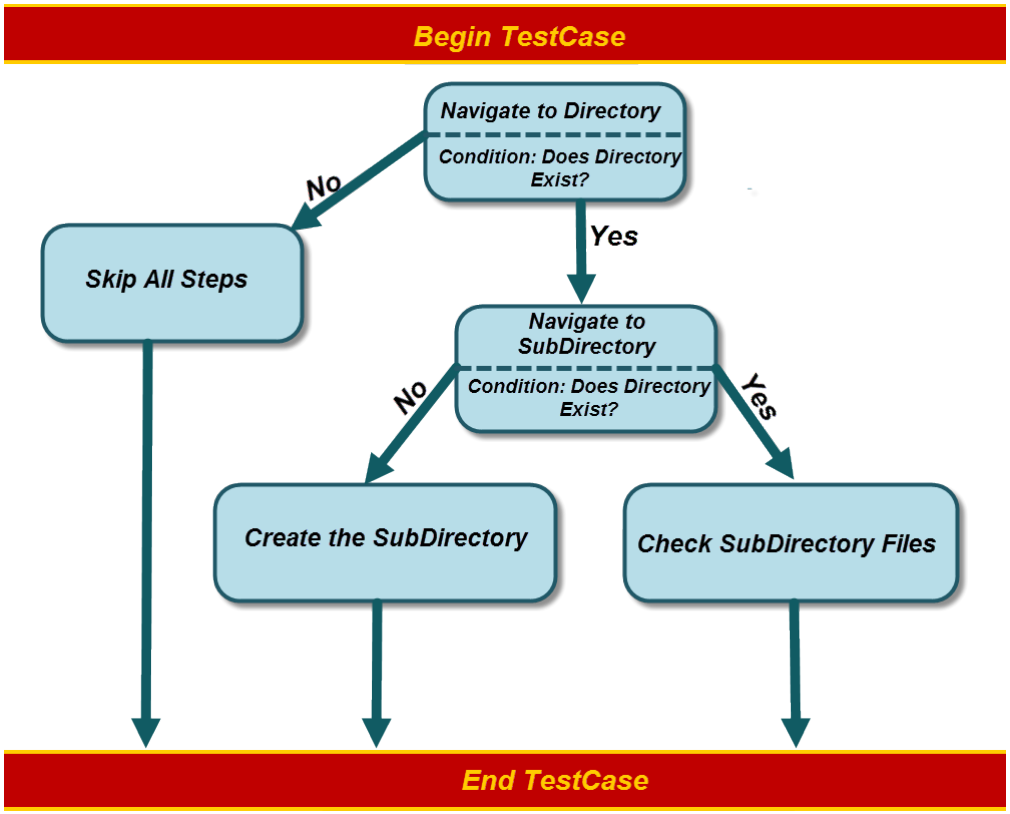
The purpose of this example **TestCase** is to evaluate files in a subdirectory on our local PC. We'll begin by connecting **TAZ** to a DOS shell. For more information on connecting equipment please see the **TAZ User Manual**.

To begin, let's visualize our **TestCase**:

:



The flowchart above clearly defines our goals for the TestCase, However it does not account for alternative possibilities. For example how should the TestCase proceed if the directories do not exist? To encompass this aspect, we will need to modify our outline:



This flowchart helps organize the conditional nature of the TestCase.

Step1 - will attempt to change the directory to:

C:\Documents and Settings\tester\AZTAZ.

If the directory does not exist, **Xpect** will fail and **Conditional{…}** will skip the remaining steps in the **TestCase**.

```
Steps:

Step1=
Send{1|cd C:\Documents and Settings\Tester\AZTAZ|DOS}
Xpect{C:\Documents and Settings\Tester\AZTAZ>}

Conditional{fail|Skip((All))}
```

Step2 - the directory was found in **Step1**. This **TestStep** now checks if the subdirectory exists. **Conditional{…}** will tell TAZ which TestStep to execute next, depending on the existence of the subdirectory.

```
Step2=
Send{1|cd C:\Documents and Settings\Tester\AZTAZ\TAZfiles|DOS}
Xpect{C:\Documents and Settings\Tester\AZTAZ\TAZfiles>}

Conditional{Pass|Run((3))|Run((4))}
```

Step3 - will run only when it is confirmed that the subdirectory exists. TAZ will then list the files in the subdirectory and **Xpect** a few important ones.

```
Step3=
Send{1|dir|DOS}
Xpect{important01}
Xpect{important03}
```

Step4 will run in the event that the subdirectory does not exist. TAZ will create the subdirectory “TAZfiles” in the directory: **C:\Documents and Settings\tester\AZTAZ**

```
Step4=
Send{1|mkdir TAZfiles|DOS}
Send{1|cd C:\Documents and Settings\Tester\AZTAZ\TAZfiles|DOS}
Xpect{C:\Documents and Settings\Tester\AZTAZ\TAZfiles>}
```

Summary

The conditional command controls the flow of a **TestCase**. It allows you to include or omit test steps based on conditions that you choose. It takes three arguments:

- ArgA: The condition which determines the action
- ArgB: the action taken if the condition is met
- ArgC: the action taken if the condition is not met

It is particularly important when using the conditional command to divide tasks into small discrete steps. Attempting to put too much in one **TestStep** will make your **TestCase** confusing and difficult to maintain. Outlining your **TestCase** or using a flow chart will save time throughout the writing process. We have discussed only a few arguments in this document. For a complete list, including many specialized arguments, please see the **TAZ Command Manual**.

Third Party Disclaimer

This disclaimer pertains to all third-parties, including but not limited to:

- Spirent Communications plc
- Ixia
- Shennick
- Ameritec
- Mozilla
- Selenium
- Net-SNMP.org
- Microsoft
- Cisco
- APC (Schneider Electric)
- ControlByWeb
- 3G Store
- Net-SNMP
- Adobe
- TestLink
- Oracle

Aztaz Software is not affiliated, associated, authorized, endorsed by, or in any way connected to any third-party or third-party products referenced in any Aztaz Software products, including but not limited to our website, software, marketing, videos, documentation or communications.

Aztaz Software does not control, endorse, or accept responsibility for any materials, products, websites, or services offered by third parties that we may reference.

All product and company names are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by their respective owners.