PIPE Test Port for TTCN-3 Toolset with TITAN, Description

Gábor Szalai

Version 1551-CNL 113 334, Rev. C, 2014-02-13

Table of Contents

Functionality

The test port establishes connection between the TTCN-3 test executor and the Unix/Linux shell.

Overview

The PIPE test port provides the following functionalities:

- Execute a given command directly without any interaction of the running command.
- Execute a given command in the background with the possibility of the live interaction of the command via `stdin`, `stdout`, and `stderr`.
- Input and output of commands can be sent as a charstring (string) or octetstring (binary data)
- Able to select method to handle new lines in the inputs and outputs
- Able to kill a background process
- Exit code of the process is returned when the process terminates
- Able to execute command attached to a new PTY.

System Requirements

In order to operate the PIPE test port the following system requirements must be satisfied:

- TITAN TTCN-3 Test Executor R7A (1.7.pl0) or higher installed. For installation guide see [2].

NOTE This version of the test port is not compatible with TITAN releases earlier than R7A.

Test Port Usage

Installation

Since the PIPE test port is used as a part of the TTCN-3 test environment this requires TTCN-3 Test Executor to be installed before any operation of the PIPE test port. For more details on the installation of TTCN-3 Test Executor see the relevant section of [2].

PIPEasp-specific compilation options to be set for building the test port:

```
LINUX_LIBS = -lxml2 -lutil
```

On a Linux machine the make file has to contain the `-lutil` flag.

Configuration

The executable test program behavior is determined via the run-time configuration file. This is a simple text file, which contains various sections (e.g. `[TESTPORT_PARAMETERS]`) after each other. The usual suffix of configuration files is *.cfg*.

The PIPE test port supports parameters as specified in the following sections.

General Test Port Parameters

- `debug`

  Set to `"YES"` if you need to debug the test port, otherwise `"NO"`.

  The default value is `"NO"`.

- `suppress_pty_echo`

  Set to `"YES"` if you need to suppress the echo of the executed command using PTY. Useful with "ssh" command.

- `auto_pid_release`

  Set to `"NO"` the execution identifier (*p_pid*) should be released by the test case explicitly.

Description of the Predefined ASPs

In this section the descriptions of the ASPs are listed.

ASP Name: `PExecute`

NOTE This ASP can be used to execute the given command with given standard input. The `PResult` ASP is sent as an answer, unless there is already a process executing which results in the ASP `PError` being sent.

This ASP can only be sent from the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PExecute {
  charstring command,
  charstring stdin
};
```

ASP Name: `PExecutePty`

NOTE This ASP can be used to execute the given command with given standard input. The `PResult` ASP is sent as an answer, unless there is already a process executing which results in the ASP `PError` being sent.

This ASP can only be sent from the test suite: DIRECTION OUT

It works in PTY mode.

Type Definition:

```
type record ASP_PExecutePty {
  charstring command,
  charstring stdin
};
```

ASP Name: `PResult`

NOTE This ASP is sent as an answer to the `PExecute` ASP. It provides information about the standard output and error of the executed command, as well as the exit code of the command.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PResult {
  charstring stdout,
  charstring stderr,
  integer    code
};
```

ASP Name: `PExecuteBinary`

NOTE This ASP is similar to the `PExecute` ASP, except that binary data is sent instead of a string as the contents of standard input. This means that the data can be, for instance, the encoded form of a PDU.

This ASP can only be sent from the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PExecuteBinary {
  charstring  command,
  octetstring stdin
};
```

ASP Name: `PExecuteBinaryPty`

NOTE This ASP is similar to the `PExecute` ASP, except that binary data is sent instead of a string as the contents of standard input. This means that the data can be, for instance, the encoded form of a PDU.

This ASP can only be sent from the test suite: DIRECTION OUT

It works in PTY mode.

Type Definition:

```
type record ASP_PExecuteBinaryPty {
  charstring  command,
  octetstring stdin
};
```

ASP Name: `PResultBinary`

NOTE This ASP is similar to the `PResult` ASP, except that the outputs are given as binary data. This ASP is sent as a result of `PExecuteBinary`.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PResultBinary {
  octetstring stdout,
  octetstring stderr,
  integer     code
};
```

ASP Name: `PExecuteBackground`

NOTE This ASP can be used to start a background process with the command given in the parameters. The `PStdin`, `PStdinBinary`, `PStdout`, `PStdoutBinary`, `PStderr`, and `PStderrBinary` ASPs can then be used to send inputs to and receive outputs from the process.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PExecuteBackground {
  charstring command
};
```

ASP Name: `PExecuteBackgroundPty`

NOTE This ASP is similar to the `ASP_PExecuteBackground`. The difference is that it executes the command with `forkpty(…)` instead of `fork(…)`

Some commands (for example ssh, scp) open a pty for user name and password instead of using stdin/stdout. The limitation of this ASP is that the stderr and stdout will be received with the same ASP: `ASP_PStdout`. If used for ssh and scp it is recommended to use `lineMode = false` because the user name and password query is sent by ssh/scp without newline.

Type Definition:

```
type record ASP_PExecuteBackgroundPty {
  charstring command
};
```

ASP Name: `PStdin`

NOTE This ASP sends input to the process started with `PExecuteBackground`. After the usage of the `PStdin` ASP, all outputs are returned to the test suite by the `PStdout` and `PStderr` ASPs.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PStdin {
  charstring stdin
};
```

ASP Name: `PStdout`

NOTE This ASP is sent to the test suite when the background process started by `PExecuteBackground` outputs something to its standard output.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PStdout {
  charstring stdout
};
```

ASP Name: `PStderr`

NOTE This ASP is sent to the test suite when the background process started by `PExecuteBackground` outputs something to its standard error.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PStderr {
  charstring stderr
};
```

ASP Name: `PStdinBinary`

NOTE This ASP is similar to the `PStdin` ASP, except that the inputs are in binary format. After sending this ASP, all the outputs produced by the background process are returned to the test suite by the `PstdoutBinary` and `PStderrBinary` ASPs.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PStdinBinary {
  octetstring stdin
};
```

ASP Name: `PStdoutBinary`

NOTE This ASP is similar to `PStdout`, except that it carries binary data.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PStdoutBinary {
  octetstring stdout
};
```

ASP Name: PStderrBinary

NOTE This ASP is similar to PStderr, except that it carries binary data.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PStderrBinary {
  octetstring stderr
};
```

ASP Name: PKill

NOTE This ASP can be used to send a KILL signal to the process started by PExecute, PExecuteBinary and PExecuteBackground. The parameter value is the signal number.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PKill {
 integer signal
};
```

ASP Name: PExit

NOTE This ASP informs the test suite about the death of the process started by PExecuteBackground. The parameter value is the exit code of the process.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PExit {
  integer code
};
```

ASP Name: PLineMode

NOTE This ASP determines the meaning of the strings representing the standard input, output, and error in the ASPs PExecute, PResult, PStdin, PStdout, and PStderr. In the first two ASPs, it determines if a newline is added to the end of the inputs and a newline is taken away from the end of the outputs. `True` determines that these changes take place, and `false` that they do not.

In the three other ASPs, `true` means that a newline is added to the end of each input string, and that the outputs are sent in separate ASPs each containing only one line of text (without the newline).

By default, the PIPE test port functions as if the PLineMode ASP would have been sent with the parameter values `true`.

NOTE If LineMode is set to `false`, the commands are not executed unless there is a newline character at the end of the input string.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PLineMode {
  boolean lineMode
};
```

ASP Name: PError

NOTE This ASP is sent to the test suite when the PIPE test port is used in a wrong manner.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_PError{
  charstring errormessage
};
```

ASP Name: PEndOfInput

NOTE This ASP closes the input to the process started with PExecuteBackground. After tis ASP is sent, no more PStdin ASP can be sent to the process. The input is closed automatically for processes started by PExecute and PExecuteBinary.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_PEndOfInput {
};
```

ASP Name: ASP_Parallel_Command

NOTE This ASP used to send the command ASPs to the test port in the case of the parallel command execution. The execution identifier (p_id) should be pre-allocated via the `f_PIPE_request_p_id` function.

This ASP can only be sent by the test suite: DIRECTION OUT

Type Definition:

```
type record ASP_Parallel_Command{
    integer        p_id,
    ASP_Commands   command
  }


  type union ASP_Commands{
        ASP_PExecute                 pexecute,
        ASP_PExecutePty              pexecutePty,
        ASP_PExecuteBinary           pexecuteBinary,
        ASP_PExecuteBinaryPty        pexecuteBinaryPty,
        ASP_PExecuteBackground       pexecuteBackground,
        ASP_PExecuteBackgroundPty    pexecuteBackgroundPty,
        ASP_PStdin                   pStdin,
        ASP_PStdinBinary             pStdinBinary,
        ASP_PKill                    pKill,
        ASP_PEndOfInput              pEndOfInput
  };
```

ASP Name: ASP_Parallel_Result

NOTE This ASP sent to the test suite by the test port in the case of the parallel command execution. The execution identifier is the same value as used in the corresponding `ASP_Parallel_Command` ASP.

This ASP can only be received by the test suite: DIRECTION IN

Type Definition:

```
type record ASP_Parrallel_Result{
  integer        p_id,
  ASP_Results    result
}

type union ASP_Results{
      ASP_PResult                    pResult,
      ASP_PResultBinary              pResultBinary,
      ASP_PStdout                    pStdout,
      ASP_PStderr                    pStderr,
      ASP_PStdoutBinary              pStdoutBinary,
      ASP_PStderrBinary              pStedrrBinary,
      ASP_PExit                      pExit,
      ASP_PError                     pError
 };
```

Function name: f_PIPE_request_p_id

NOTE This function should be used to allocate a new execution identifier for parallel command execution (ASP_Parallel_Command and ASP_Parallel_Result). The allocated identifier valid until the final result will be received by the test suite, except reusable identifier was allocated by setting the "reuse" parameter to "true". In that case, the identifier can be used to start a new command.

Type Definition:

```
external function f_PIPE_request_p_id(
        inout PIPEasp_PT pl_port,
        in boolean pl_reusable:=false) return integer
```

Function name: f_PIPE_release_p_id

This function should be used to release the execution identifier allocated by the f_PIPE_request_p_id. If the execution identifier is not reusable and the auto_pid_release is not set to "NO", the execution identifier released by the test port when the ASP_PExit has been inserted into the port queue.

Type Definition:

```
external function f_PIPE_release_p_id(
  inout PIPEasp_PT pl_port, in integer pl_pid)
  return boolean;
```

Exit status handling functions

The following functions are designed to process the exit status of the process. These functions have the same functionality as the `POSIX` compliant `WIFEXITED`, `WEXITSTATUS`, `WIFSIGNALED`, `WTERMSIG` macros.

```
external function f_PIPE_WIFEXITED(in integer code) return boolean;
external function f_PIPE_WEXITSTATUS(in integer code) return integer;
external function f_PIPE_WIFSIGNALED(in integer code) return boolean;
external function f_PIPE_WTERMSIG(in integer code) return integer;
```

Error messages

The following list contains the error messages of the PIPE testport:

`select system call fails (<errno>): <reason>`

This is a critical error. The error number and the description are shown.

`Cannot redirect stdin/stdout/stderr`

The stdin, stdout or stderr cannot be redirected, because dup2 system call failed

Warning messages

The following list contains the possible warning messages of the PIPE test port:

`Unexpected message from stdout/stderr, no command is executing`

Message received on stdout/stderr, but there is no process executing.

Examples

There is a simple example test case in the demo directory. It contains the following files:

***PipeTest.ttcn*, *PIPEasp_Templates.ttcn*, *ShellNotice.sh*, *ShellQuestionString.sh*, *ShellQuestionStringYesNo.sh*, PipeTest.**

These files define the example test suite and some useful templates. The example test suite can be used as a starting point when using the PIPE test port. There is also an associated configuration file *PIPE.cfg*, which can be used to execute the test suite. There are also three shell scripts that can be added to any test suite and used for presenting notices to the user, or for asking things from the user. The notices and questions are presented in their own X window.

Before running the test case set the *ShellTestDir* environment variable to the demo directory.

The file *PipeTest.prj* is the project definition file for the TITAN GUI.

Terminology

No specific terminology is used.

References

[1] ETSI ES 201 873-1 v4.3.1 (2011-06) The Testing and Test Control Notation version 3. Part 1: Core Language

[2] User Guide for TITAN TTCN–3 Test Executor

[3] Programmer's Technical Reference for TITAN TTCN–3 Test Executor