Terminal Agnostic Automation - Native Terminals

Introduction

This will provide information about how to automate a Windows Native Mainframe Terminal for which you don't have the means to establish a Terminal Session via EHLL API or directly or for any other difficult situation.

This method should be preferred over current Internal Provider (June 2018) if that support does not improve stabilize as it is more reliable.

There will be a step-by-step guide that will explain which generic activities to use.

Note: If you find the whole thing "I already know this" boring, please at least check out the "Data Output Methods" section, that's kind of the highlight for all this and it's one of the things where I haven't seen many using all the features..

Advantages

Well, this is your safety net.

It is as generic as it can get and could serve in as many situations as there can be.

For as long as you deal with a Desktop App and not a Java based terminal (by the way, the agnostic terminals web guide should be very similar to this, once I get some test Web based terminal to prove it) - this should serve you well.

There are no significant performance issues (in some cases this is faster than the actual direct provider) so there is no reason to fear it. We might document some performance tests so you have the convincing numbers as well.

This really is a solution, not a workaround, and it's a better solution than using Internal Provider (which it seems to be the faulty safety net of choice right now).

Disadvantages

This is agnostic (generic). You won't be getting detailed error messages as you do when using EHLL interface, it will be harder to understand what happened when something happens, you'll need to work it out when you get errors. EHLL told you exactly what happened with the terminal itself as well (if you read the terminal log), now you'll need to check and try out stuff.

You don't have a session and a connection scope anymore, but Terminals will still disconnect on timeouts, network issues (those are external issues, not UiPath issues). So to do this right, you will still need to implement a similar approach with this one: EHLL Recoverable Terminal Session and Connection - let me know if there are issues adapting that for this approach.

Also, since this flow is not contained within a scope, bad development will mix stuff in more confusing ways than ever before, so try to keep it at clean and as contained as possible to allow making sense of what's happening (by others that might need to troubleshoot later on).

Don't make a Terminal Session at all

- "Attach Window" Activity

You reached this point because establishing a Terminal Session connection failed or because this method is more viable than that.

So don't open a Terminal Session at all (and clean previous attempts).

Use an Attach Window activity and indicate the Terminal Emulator Window

Positioning in the Terminal Emulator screen

- "Send Hotkey" Activity with "TAB"

- "Send Hotkey" Activity with "Shift+TAB"

This is the most reliable way to navigate the Terminal Emulator screen.

TAB moves to the next input field in the screen..

Shift+TAB moves to the previous input field in the screen.

When using "Indicate on screen", just point it to the Terminal Screen, do not panic because you can't fine tune it on a smaller element, it will work.

Data Input Methods

- "Type Into" Activity

- "Send Hotkey" Activity

You should already know what "Type Into" does and how it replaces "Set Field" or "Send Keys" terminal activities. For "Set Field At Position" terminal activity, you will need to navigate to the right position first, so use TABs and Shift+TAB for that (see **Positioning in the Terminal Emulator screen** section right above this one) and then use "Type Into".

"Send Hotkey" activity is a replacement for "Send Control Key" terminal activity. There is one important difference here:

- "Send Control Key" takes the AID function as a parameter (such as "TRANSMIT" or "PF3", or "PF14" - these are the original terminal keyboard keys by the way)
- "Send Hotkey" takes the actual PC key as a parameter (for "TRANSMIT" it can be "Enter" or "Right-Ctrl" - you need to check as it varies from Terminal Emulator to Terminal Emulator. And for "PF3" AID key, probably you need to send "F3", for "PF14" maybe "Shift+F2").
- There should be a keyboard mapping in Terminal Emulator's configuration that gives you the whole correspondence between PC Keyboard and legacy Terminal Keyboard). You can also ask the user "What do I press here?" and he'll probably say "Shift+F2" and not "PF14" (and that's exactly what you need).

Data Output Methods

- "Screen Scrape" Activity with "Clipping Region" and "Generate Table" for Tabular Data (structured, tables)

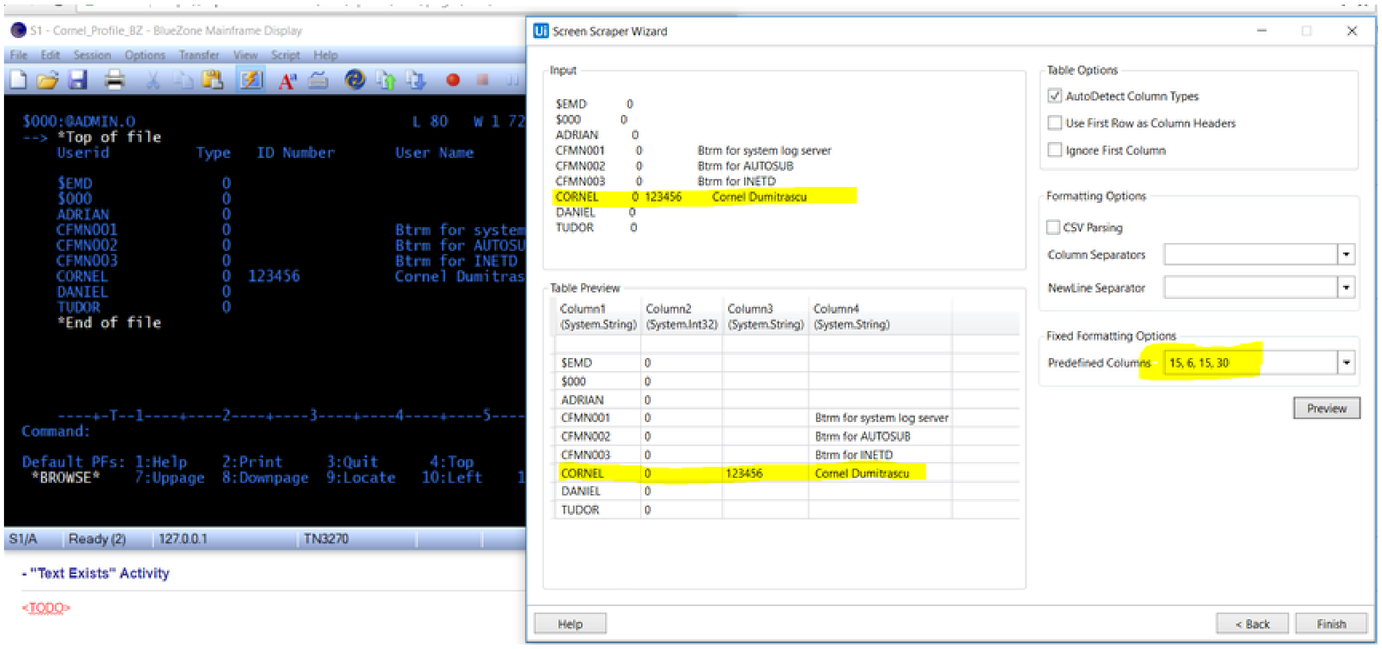
- "Get Visible Text" Activity with "Clipping Region" for unstructured data

- "Get Visible Text" Activity without Clipping Region for reading the entire screen

Reading structured data as DataTable:

It involves using "Screen Scraping" with "Clipping Region", "Generate Table" and "Fixed Formatting Options". It sounds like a lot, but it's not.

Here's a screenshot of how it looks like:


S1 - Cornel_Profile_BZ - BlueZone Mainframe Display
File Edit Session Options Transfer View Script Help
S1/A Ready (2) 127.0.0.1 TN3270
- "Text Exists" Activity
<TODO>
Screen Scraper Wizard
Help
< Back
Finish
Preview

```
$000:@ADMIN.O                                     L 80   W 1 72
--> *Top of file
    Userid          Type   ID Number        User Name

    $EMD               0
    $000               0
    ADRIAN             0
    CFMN001            0                    Btrm for system
    CFMN002            0                    Btrm for AUTOSU
    CFMN003            0                    Btrm for INETD
    CORNEL             0  123456            Cornel Dumitras
    DANIEL             0
    TUDOR              0
    *End of file

    ----+-T--1----+----2----+----3----+----4----+----5----
Command:

Default PFs: 1:Help     2:Print     3:Quit      4:Top
 *BROWSE*    7:Uppage   8:Downpage  9:Locate   10:Left     1
```

Input

```
$EMD        0
$000        0
ADRIAN      0
CFMN001     0          Btrm for system log server
CFMN002     0          Btrm for AUTOSUB
CFMN003     0          Btrm for INETD
CORNEL      0 123456       Cornel Dumitrascu
DANIEL      0
TUDOR       0
```

Table Preview

Column1 (System.String)	Column2 (System.Int32)	Column3 (System.String)	Column4 (System.String)
$EMD	0		
$000	0		
ADRIAN	0		
CFMN001	0		Btrm for system log server
CFMN002	0		Btrm for AUTOSUB
CFMN003	0		Btrm for INETD
CORNEL	0	123456	Cornel Dumitrascu
DANIEL	0		
TUDOR	0		

Table Options
- [✓] AutoDetect Column Types
- [ ] Use First Row as Column Headers
- [ ] Ignore First Column

Formatting Options
- [ ] CSV Parsing
- Column Separators:
- NewLine Separator:

Fixed Formatting Options
- Predefined Columns: 15, 6, 15, 30

It's neat, it works, it's straightforward, I find it to be the best way to do this, please use it.

Reading unstructured data:

Use "Get Visible Text" with "Clipping Region" (press F3 when after clicking on "Indicate element" and then define the area you want to read).

Reading the entire screen:

This is the easiest. Use "Get Visible Text" activity with default options and point at Terminal screen.

You get the screen as a big String and then you can use String operations or Regular Expressions to do whatever with it.

Screen Transition Methods

- "Text Exists" Activity

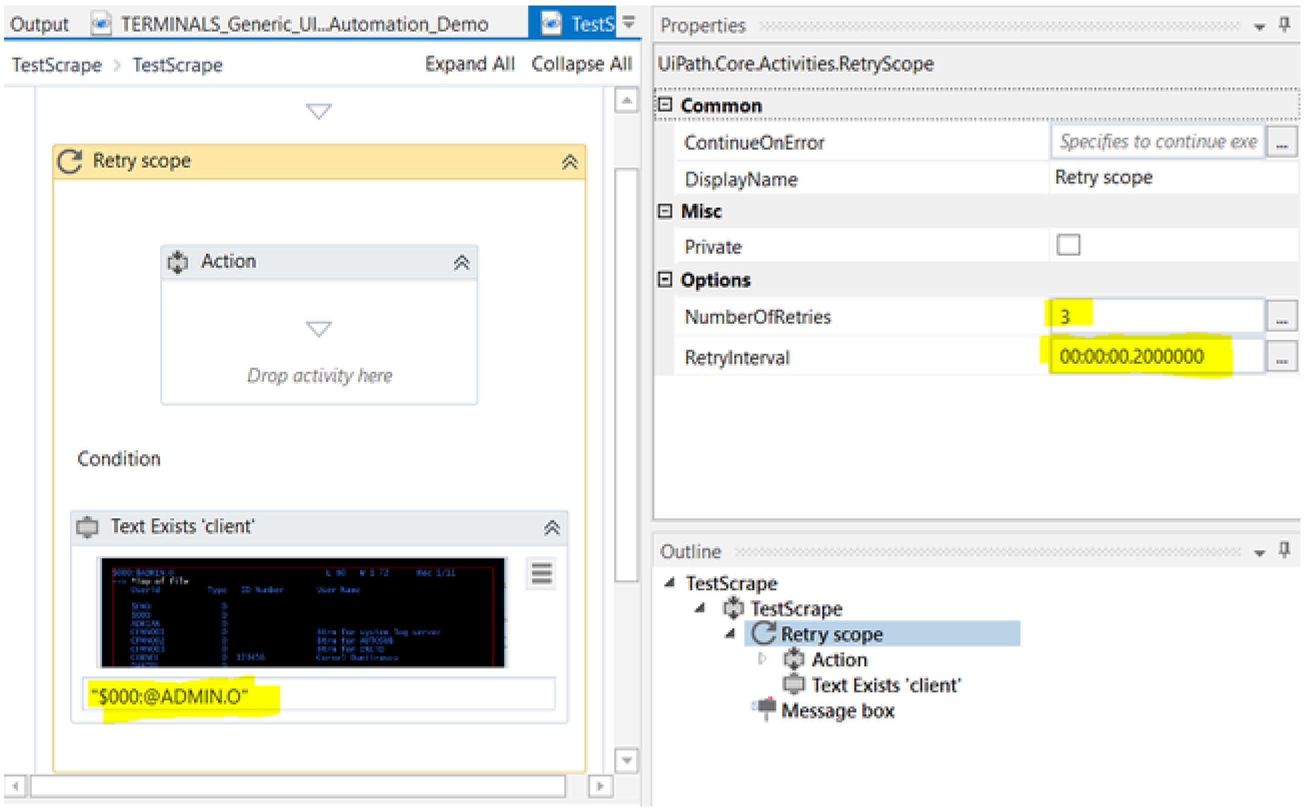
You probably won't need this all the time, but when you do, here's how:

When working with Terminal Activities you had the "Wait Text At Position" and "Wait Screen Text" activities to handle a screen switch (at least I hope so, for using DelayMS property to handle screen transition is slow, inaccurate, dangerous).

Now you need to use "Text Exists" Activity that does just about the same thing.

The one catch is that you might need to use "Text Exists" in a Retry Scope, because "Text Exists" is just a check, does not really wait, so you control the actual wait from Retry scope (and "Text Exists" should be the condition for the Retry scope).

In a Screen A to a Screen B transition, for "Text Exists" you should provide a relevant text from Screen B that distinguishes it from screen A.



"$000:@ADMIN.O"

Properties

UiPath.Core.Activities.RetryScope

Common	
ContinueOnError	*Specifies to continue exe*
DisplayName	Retry scope
Misc	
Private	☐
Options	
NumberOfRetries	3
RetryInterval	00:00:00.2000000

Outline

- TestScrape
  - TestScrape
    - Retry scope
      - Action
      - Text Exists 'client'
    - Message box