YAT

Yet Another Terminal

See screenshots at https://sourceforge.net/projects/y-a-terminal for hints on features and available settings.

Quick Reference

Send Format

Radix	*Escape*	*Example*	*Remarks*
String	 \s(…)	OK \s(OK)	Default, no escape needed. [Explicit Default Radix] to change default.
Character	\c(…)	\c(O)\c(K) \c(O K)	Use \\ to send a backslash '\'. Use \) to send a closing parenthesis ')'.
Unicode	\U+ \U(…) \u	\U+20AC \U(20AC) \u20AC	 C-style.
ASCII mnemonic	<…>	<CR><LF> <TAB>	Use \< to send an opening angle bracket '<'.
Hexadecimal	\h(…) \0x… \x…	\h(4F 4B) \h(4F4B) \0x4F\0x4B \x4F\x4B	 C-style.
Decimal	\d(…)	\d(79 75)	
Octal	\o(…) \0…	\o(117)\o(113) \0117\0113	
Binary	\b(…) \0b…	\b(01001111) \0b01001111	

Note the [Enable <…> and \… Escapes on [Send Text|File]] setting
available in context menu and [Terminal → Settings... → Advanced… → Send].

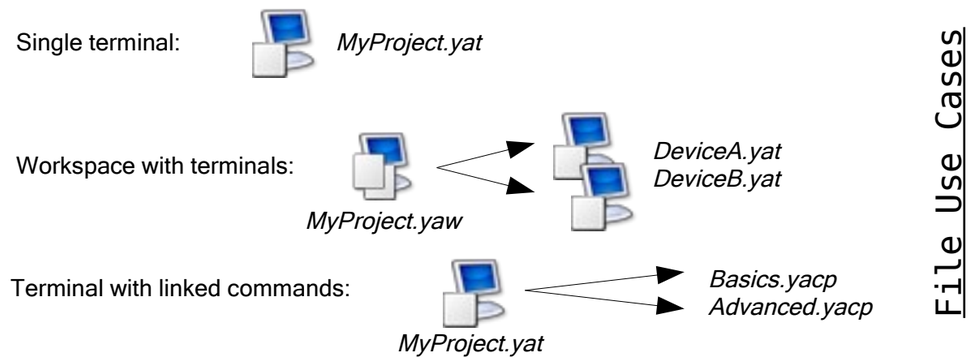
Keywords

Text	*Remarks*
Send something, then clear monitor\!(Clear)	
Send something\!(Delay(1000), then send delayed Send a line, then delay before the next line\!(LineDelay(1000)) Send a line, then delay before the next line\!(LineInterval(1000)) Send a line, then delay before the next line\!(LineInterval)	Delay 1000 ms within line. Delay 1000 ms at the end of the line, after EOL. Delay such a line interval of 1000 ms results. Delay according to value in advanced settings.
Send something multiple times\!(LineRepeat(10)) Send something infinitely\!(LineRepeat(-1))\!(LineInterval(1000))	Repeat 10 times, as fast as possible. Repeat infinitely, with a line interval of 1000 ms.
Send content without appending EOL\!(NoEOL)	Suppress appending EOL (End-Of-line).
Send a line\!(EOL), then send another line	Inject additional EOL.
Send time information, e.g. now = \!(TimeStamp)	Inject time stamp. Format according to [View → Format Settings...].
\!(Port(10))	On-the-fly change of serial COM port.
\!(PortSettings(19200, 7, 1))\!(Baud(19200))\!(FlowControl(1))	On-the-fly change of serial port settings.
\!(RtsOn|Off|Toggle) \!(DtrOn|Off|Toggle)	On-the-fly change of serial port control signals.
\!(FramingErrorsOn|Off|Restore)	Serial COM port error handling.
\!(OutputBreakOn|Off|Toggle)	Serial COM port break handling.
\!(ReportID(1))	On-the-fly change of USB Ser/HID report ID.

Actions and Shortcuts

Scope	*Action*	*Shortcut*	*Action*	*Shortcut*
Terminal	Edit Text	Alt+E	Send Text Send Text w/o EOL	F3 Ctrl+F3
	Toggle Keep Text	Ctrl+Shift+K	Toggle Send Imme.	Ctrl+Shift+I
	Select File	Alt+I	Send File	F4
	Open/Start Close/Stop	Ctrl+Shift+O Ctrl+Shift+C	Break Operation	Ctrl+B
	Toggle Formatting	Ctrl+Shift+T	Settings...	Ctrl+Shift+S
Predefined	Page	Ctrl+<i>	Define...	Ctrl+Shift+D
	Page Next Page Previous	Ctrl+Alt+→ Ctrl+Alt+←	Send Copy	Shift+F<i> Ctrl+Shift+F<i>
Monitor	Clear All Terminals Clear	Ctrl+L Ctrl+E	Refresh All Terminals Refresh	F5 Ctrl+F5
	Select All Select None	Ctrl+A Ctrl+N	Copy to Clipboard Save to File...	Ctrl+C Ctrl+T
	Find	Ctrl+F	Print...	Ctrl+P
Find	Case Sensitive Whole Word Regular Express.	Alt+C Alt+W Alt+E	Next Previous All	Alt[+Shift]+N Alt[+Shift]+P Alt[+Shift]+L
Log	On Off	Ctrl+Shift+N Ctrl+Shift+F	Settings...	Ctrl+Shift+L
Window	Automatic Layout Tile Horizontal Tile Vertical	Alt+Shift+A Alt+Shift+H Alt+Shift+V	Cascade Minimze Maximize	Alt+Shift+C Alt+Shift+I Alt+Shift+X
Workspace	New Terminal… Open Terminal	Ctrl+N Ctrl+O	Save Terminal Save All Terminals Save Workspace	Ctrl+S Ctrl+Shift+A Ctrl+Shift+W
	Close Terminal	Ctrl+F4	Toggle Terminals	Ctrl+TAB
Application	Exit	Alt+F4	Always On Top	Alt+Shift+T
Help	Contents...	F1		

File Use Cases



Quick Reference 1.0 :: 2021-03-25
YAT 2.4.1 :: and above

Quick Reference

See [Albatros-UserDocumentation.pdf] for introduction, HowTo's and best practices.
See [Albatros-ScriptInterfaceReference.chm] for details and complete list of methods.
Search contents of [AlbatrosScript.zip] for usage examples and additional information.

Albatros

YAT with Scripting

`Send` triggers a send request, the actual sending happens asynchronously.
`Receive` handles available message(s); or waits for message(s) until a 'ReceiveTimeout' occurs.

Method Group	*Functionality*
`Send[Multiple]`	Sends [multiple] text including the configured EOL sequence. Does not wait for a response.
`[Send[Multiple]And]Receive`	Returns the available message(s); or waits for message(s) until a 'ReceiveTimeout' occurs.
`[Send[Multiple]And]ReceiveAll`	Receives all messages until a 'ReceiveTimeout' occurs.
`[Send[Multiple]And]ReceiveMatch[es]`	Matches the available message(s); or waits for message(s) until a 'ReceiveTimeout' occurs.
`[Send[Multiple]And]Receive[Matches]For`	Receives messages for the specified duration.
`[Send[Multiple]And]Receive[Matches]Until`	Receives messages until the specified terminating message.
`[Send[Multiple]And]Receive[Matches]UntilWithin`	Receives until the terminating message within the specified interval.
`[Send[Multiple]And]Receive[Match[es]]Within`	Expects message(s) within the specified interval.
`SendFor[AndReceiveAll]`	Sends text for the specified duration.
`SendWithoutEOL`	Sends text without appending the EOL sequence.
`SendXOn|XOff`	Sends an XOn|XOff control byte.

Examples:

```
void SendAndReceive(       string send,                              out string received);
void SendAndReceiveMatch(  string send,             string expected,   out string received);
void SendAndReceiveMatch(  string send,              Regex expected,   out string received,   out Match receivedMatch);
void SendAndReceiveMatches(string send, IEnumerable<Regex> expected, out string[] received, out Match[] receivedMatches);
```

Methods provide overloads for `string` as well as `Regex` and `Match` objects.
Methods provide overloads omitting options as well as `out` parameters.
Methods are available with the **implicit = active terminal** as well as **explicit terminal** programming model:

```
void SendAndReceiveMatch(                string send, string expected);
void SendAndReceiveMatch(int terminalId, string send, string expected);
```

All methods are available with **exception handling** as well as **success flag** result indication:

```
void    SendAndReceiveMatch(string send, string expected);
bool TrySendAndReceiveMatch(string send, string expected);
```

IConnection

Method/Property	*Functionality*
`void Open();` `void Close();`	Opens/starts the connection of the terminal. Closes/stops the connection of the terminal.

`bool IsStarted();`
`bool IsOpen();`
`void WaitUntilIsOpen();`
	Returns whether the connection has been started or is open.
	Waits until the connection has opened.

`void GetActiveSettings(`
`  SerialSettings  settings);`
`  NetworkSettings settings);`
`  USBSettings  settings);`
	Gets the settings of the connection.

`void ChangeActiveSettings(`
`  SerialSettings  settings);`
`  NetworkSettings settings);`
`  USBSettings  settings);`
	Changes the settings of the connection.

`void Break();`	Breaks all currently ongoing operations, e.g. sending a file.

`bool IsSending();`
`bool IsReceiving();`
`void WaitWhileIsSending();`
`void WaitWhileIsReceiving();`
	Indicates whether sending or receiving is currently ongoing.
	Waits while sending or receiving is ongoing, i.e. until 'IsSending()' or 'IsReceiving()' no longer returns 'true'.

`bool HasAvailable();`
`int GetAvailableCount();`
`int PeekAvailable(out string available);`
	Get information on the messages available for the script.

`int ClearAvailable();`	Clears all messages available.
`bool AutoClearAvailableOnSend { get; set; }`	Configures automatic clear of available messages on send.
`int ReceiveTimeout { get; set; }` `void ResetReceiveTimeout();`	The time until a receive time-out occurs, in milliseconds. 0 to switch time-out off. Reset to 'DefaultReceiveTimeout'.
`bool AutoScriptIOLog { get; set; }`	Configures logging of the script's send and receive calls.

ITerminal

Method/Property	*Functionality*
`int ActiveId { get; }`	Gets the ID of the active terminal.
`int[] AvailableIds { get; }`	Get the IDs of the available terminals.
`int AvailableCount { get; }`	Get the number of the available terminals;.

`void Open(string portName);`
`void Open(SerialPortSettings  settings);`
`void Open(TCPIPClientSettings  settings);`
`void Open(TCPIPServerSettings  settings);`
`void Open(TCPIPAutoSocketSettings settings);`
`void Open(USBSerHIDSettings  settings);`
	Creates or opens a terminal with given settings.

`void GetActiveSettings(out TerminalSettings settings);`	Gets the settings of the terminal.
`void ChangeActiveSettings(TerminalSettings settings);`	Changes the settings of the terminal.
`void Close();`	Closes the terminal.

IScriptControl

Method/Property	*Functionality*
`void Wait(int milliseconds);` `void Wait(TimeSpan timeout);`	Suspends the script for the specified time.
`bool ControlFlag { get; set; }`	Lets the user control the script.
`bool StopOnErrorFlag { get; set; }`	Lets the user configure whether to stop the script in case of an error; or continue.
`void Exit();`	Exits the script immediately.

ILog

Method/Property/Type	*Functionality*

`void Create();`
`void Create(string scriptRevision);`
`void Create(string scriptRevision, string fileNameOrPath);`
`void Create(string fileNameOrPath, bool appendTimeStampToFileName);`
	Creates a log file.

`void WriteLine(LogLevel logLevel, string message);` `void WriteLine(LogLevel logLevel, string format, params object[] args);`	Log a (formatted) text line.
`LogLevel { Fatal = 1, Error = 10, Warning = 100,` `  Info = 1000, DbgInfo = 10000 }`	Log level values.
`LogLevel LogLevelThreshold { get; set; }`	The level that shall be logged.
`bool ForwardLogDataToScriptEngineDialog { get; set; }`	Log content is also shown to user.
`void Close(int result);`	Closes the log file.

IUser

`void WriteLine(string message);` `void WriteLine(string format, params object[] args);`	Write a (formatted) text line.
`void ClearOutput();`	Clears the user output.
`void Beep();`	Plays a system beep.
`double InputValue(string message);`	Requests a value from user.
`string InputText(string message);`	Requests a text from user.

CS-Script

`//css_import .\SomeMoreCode.cs;`	Import code files, same as `#include` in C/C++.
`//css_reference .\SomeAssembly.dll;`	Refer to additional .NET assemblies.
`//css_resource .\SomeForm.resx, .\Project.SomeForm.resources;`	Handle Visual Studio Designer generated resources.
`//css_searchdir ..\Assemblies;`	Alternative to using relative paths in directives above.

Quick Reference 1.0 :: 2021-03-25
Albatros 1.28.01 :: and above