wait.mds library for supporting automation based on MS-DOS scripts.

The library is dedicated to the memory
of Nikolai and Raisa Kopiev.

Introduction.

The wait.mds library is a tool of automation support based on MS-DOS scripts. The entire spectra of library functionality is entirely contained in the file "wait.mds.bat", which can be used as an executable file for calling it from any local folder, as well as an installer for installing and registering the library.

The library functionality is divided into external calls to internal procedures and event handling. The library also contains macro definitions (macros) that can be loaded into the context of the user's batch file for execution.

Because the library's functionality is mainly intended to solve problems of automating certain actions on a local computer, it is intended for use on servers running Windows OS or for solving server problems on a local computer. For correct operation, it is recommended to use the library under a user with administrator rights.

Library Installer.

The library file "wait.mds.bat" for its use generally does not require any pre-installation of software on Windows OS of all versions from XP to 11 and can be used as is on any machine. As the user calls the library functionality elements, the required installation are made automatically in background. The exception is the required pre-installation of Microsoft .NET Framework 3.5 SP1 on Windows XP, because this OS does not have a pre-installed framework. Also, the OS security system must be configured for console applications to work. If these settings were not made, then they should either be made manually or the installer should be run for automatic configuration:

call wait.mds.bat /sub:install /install /all

After manual or automatic configuration of settings, it is necessary to reboot computer for them to take effect. If you are installing, you will need to run it again after rebooting to complete it.

As a result of the full installation, the subdirectory "wait.mds" will be created in the %ProgramFiles% folder, which will contain: a copy of the original library file; a lightweight version of the library in the file "wait.mds.lite.bat"; a web help file; a text file in console encoding with performance counters for the utility "typeperf.exe"; compiled and registered COM server files (x86/x64 depending on the bitness of the OS); the source code of the COM server in VB.NET and an auxiliary file in VBScript from the installer. Also, the environment variable %wait.mds% is registered to run the library "wait.mds.lite.bat" from any folder and a shortcut is created on the desktop for the web help file, which can also be opened using the keys "Ctrl-Alt-Shift-W".

In addition to the "/all" key, it is also possible to install only the COM server with the "/vb" key, install only the performance counters with the "/tpc" key, and also install with the "/lib" key, which performs all the actions from the full installation, except for the actions during installations with two previous keys.

Calls of internal procedures.

External call of internal library procedures is implemented through a formatted first argument string, containing the prefix substring "/sub:" and a suffix substring containing name of procedure. The following arguments are arguments of internal procedures. The main procedures are the installer procedure, as well as the procedures "importMacros" and "unsetMacros" for loading and unloading macro definitions from the context of the user's batch file. The installed version of the library "wait.mds.lite.bat" does not have an installer procedure. A full description of external procedures is given in the web help file.

Event handling.

Event handling is a library call with parameters that indicate that the batch file execution should be stopped until a certain change in the system is detected, after which it should continue run. A full description is given in the web help file, a short list of the types of events that can be monitored: appearance or disappearance of a file or folder; change in file or folder attributes; process start or exit; appearance or closing of a window with a title; change in network traffic; change in read/write intensity; change in free disk space; change in CPU load; change in process activity; change in window state (maximized/minimized, etc.); waiting for the specified text to appear in the console; waiting for the specified value of an environment variable in another console. A list of events from an external file in JSON format and custom events that are defined by a user-developed macro are also supported.

Importable Macro Definitions.

Macro definitions are imported into the user's batch file context using a library external procedure. Their full description is provided in the web help file that is generated by the installer. List of library macro definitions:

1.	@errorLevel	- sets the error level to a digital value in the user context;
2.	@isok	- compliance of imported macro with the delayed expansions state;
3.	@exit	- exits the call stack of batch files and procedures (labels);
4.	@error	- reports an error and terminates the script run, manageable exit;
5.	@echo_imprint	- prints text into the same console line without caret return;
6.	@imprintxy	- prints text with shift of the row and the column and returns caret;
7.	@drop_conrows	- clears the specified number of last console lines;
8.	@mac_wrapper	- runs a macro and returns its result;
9.	@mac_wraperc	- runs a macro and returns its result (it needs definitions cleanup);
10.	@mac_check @istrue	- runs macro, returns its result and sets error level;
11.	@spinner	- runs a loop until the specified timeout in milliseconds;
12.	@mac_spinner	- runs a loop until the macro completes and exits the loop;
13.	@mac_loop @mac_do	- runs a loop until the macro completes, has loop body;
14.	@unset	- removes variable definitions by clearing their values;
15.	@unset_mac	- removes macro definitions from local environment;
16.	@unset_alev	- removes environment variables;
17.	@runapp	- starts a new application using the `start` command;
18.	@chcp_file	- recodes the text of a file from one code page to another;
19.	@str_length	- calculates the length of a string;
20.	@str_unquote	- removes all quotes in a string or replaces them;
21.	@fixedpath	- searches for and gets the full path of an object on disk;
22.	@shortpath	- converts a disk object's path string to a DOS "8.3" name format;
23.	@get_number	- sets the numeric value of a variable;
24.	@get_xnumber	- sets the numeric value of a variable (x16 -> x10);
25.	@rand	- generates a random number in a given range;
26.	@echo_params	- displays a list of parameters;
27.	@enumA @enumB	- prints the result of 'for-in-do' one after another;
28.	@ipaddress	- returns local IP address;
29.	@web_avail	- checks if internet connection is available;
30.	@web_ip	- returns the local IP address of the internet connection;
31.	@regvalue	- gets value from Windows registry in console code page;
32.	@shellfolder	- returns full paths to special folders;
33.	@oemtocp	- encoding from OEM Windows to console code page;
34.	@cptooem	- recoding from console code page to OEM Windows;
35.	@str_decode	- replaces pseudo tags with codes to string characters;
36.	@syms_replace	- replaces all occurrences of characters with other characters;
37.	@sym_replace	- replaces all occurrences of a character with another character;
38.	@syms_cutstr	- gets substrings between character(s);
39.	@pid_title	- returns the window title of the process with the given PID;

40.	@title	- gets the window title or process identifier (PID);
41.	@title_pid	- finds the PID of a window using a substring of its title;
42.	@substr_remove	- removes substrings inside delimiters of string;
43.	@substr_get	- outputs strings from substrings inside delimiters;
44.	@substr_extract	- outputs substrings within specified delimiters;
45.	@substr_regex	- substring matching regular expression;
46.	@str_trim	- clears leading and trailing characters;
47.	@str_encode	- replacing control characters with their ASCII codes;
48.	@str_clean	- removes selected sets of characters from a string;
49.	@str_plate	- overwrites selected sets of characters by character;
50.	@str_arrange	- reorganization of substrings in a string;
51.	@str_upper	- converts a string to uppercase letters;
52.	@str_isempty	- checks if the parameter is empty;
53.	@date_span	- gets time range using WMIC format of date and time;
54.	@time_span	- extracts time values from a time command;
55.	@obj_attrib	- reads attributes of a file or folder on disk, can changes them;
56.	@obj_size	- obtains size of a file or folder on disk;
57.	@disk_space	- returns total, free and used disk space;
58.	@exist	- checks for the presence of an object on disk;
59.	@exist_check	- the object's presence along with the health status of the disk;
60.	@obj_newname	- generates new object's name in specified location;
61.	@perf_counter	- gets localized typeperf.exe counter;
62.	@typeperf	- returns the result of the typeperf.exe query.
63.	@typeperf_devs	- returns a list of devices from a typeperf.exe query;
64.	@typeperf_res_a	- determines the usage of the device by two counters;
65.	@typeperf_res_b	- determines the use of the device;
66.	@typeperf_res_c	- determines the usage of the device by two counters (optimized);
67.	@typeperf_res_d	- determines the use of the device (optimized);
68.	@typeperf_res_use	- returns the usage of a device by its hardcoded type identifier;
69.	@nicconfig	- finds a network device and returns its settings;
70.	@netdevs	- returns CSV lists of basic properties of network devices;
71.	@res_select	- performs a task and selects elements of its result;
72.	@event_file	- adds or removes an entry from the event file;
73.	@event_item	- gets the specified event attribute with id;
74.	@runapp_getpid	- launches the application and returns its PID;
75.	@sleep_wsh	- stop (sleep) the calling process for a timeout in milliseconds;
76.	@runapp_wsh	- launching an application using the `WScript.Shell` object;
77.	@sendkeys	- send keyboard keystrokes to the active window;
78.	@shortcut	- creates a Windows shortcut of the specified object;
79.	@hex	- converts a decimal number to a hexadecimal string;
80.	@taskinfo	- returns information about the process;
81.	@procpriority	- reads or sets the priority of a process;
82.	@findwindow	- finds a window using the window class name and its title;
83.	@windowstate	- checks the state of the specified window attribute;
84.	@activewindow	- returns the handle of the currently active window;
85.	@foregroundwindow	- returns the handle of the foreground window;
86.	@windowclass	- returns the class of the window with the specified handle;
87.	@windowcaptext	- returns the title of the window with the specified handle;
88.	@windowrect	- returns the client area rectangle of the window;
89.	@showwindow	- shows a window according to the given command;
90.	@findshow	- finds a window by class and title names, shows it;
91.	@movewindow	- moves the window to a point in absolute coordinates;

92.	@sendmessage	- sends a message to the window with the specified handle;
93.	@findcontrol	- finds a child window of a parent window;
94.	@closewindow	- closes or minimizes the window;
95.	@showdesktop	- macro minimizes all windows and shows the desktop;
96.	@repaint	- redraws the specified window or desktop;
97.	@windowsofpid	- returns window handles of the process with the specified PID;
98.	@pidofwindow	- returns the PID of the process of the window with the handle;
99.	@coprocess	- returns the PID of the child or parent process;
100.	@screenshot	- takes a screenshot of a window or screen and saves it to a file;
101.	@compareshots	- byte by byte comparison of two screenshots;
102.	@cursorpos	- returns the absolute coordinates of the mouse pointer;
103.	@mouseclick	- emulates a mouse button press;
104.	@monitor	- returns the monitor ID;
105.	@appbarect	- returns the Windows taskbar rectangle;
106.	@screenrect	- returns the rectangle of the monitor's client area;
107.	@screensize	- gets or changes monitor screen resolution;
108.	@movetoscreen	- moves the window to the client area of the monitor;
109.	@consoletext	- reads the text of the specified console window;
110.	@shrink	- data compression with string representation of result;
111.	@comparefiles	- performs a byte-by-byte comparison of data in two files;
112.	@environset	- prints all environment variables of the console process;
113.	@enwalue	- gets environment variable value from process `cmd.exe`;
114.	@radix	- converts numbers from one number system to another;
115.	@code	- converts a string into a sequence of hexadecimal ASCII codes;
116.	@callstack	- returns the current call stack in the specified `cmd.exe`.

Set of additional files for the library.

Additional files have been added to the wait.mds library to better meet the criteria for open source software, to document the components being installed, and, where necessary, to make it easier to work with its source code. For instance, an expanded macro definition can be temporarily moved to a local user file for tracing or used to examine library source code. The list of files with their relative locations is:

1. ".\docs\Назначение и сигнатуры COM-функций библиотеки wait.mds.pdf" - description of the COM server in Russian;
2. ".\docs\Purpose and signatures of COM functions of the wait.mds library.pdf" - description of COM server in English;
3. ".\macros\macros.general.collapsed.bat" - complete list of collapsed definitions of general purpose macros;
4. ".\macros\macros.general.expanded.bat" - complete list of expanded definitions of general purpose macros;
5. ".\macros\macros.special.expanded.bat" - a complete list of macro definitions that are used inside the library, as well as macro definitions that are represented as recycled strings inside general-purpose macros;
6. ".\tests\wait.mds.tests.macros.bat" - a complete suite of tests for general-purpose macros. In addition to listing macro tests, it also lists examples of their use;
7. ".\tests\call.stack.test\.." - subfolder used by tests of general purpose macro;
8. ".\tests\call.stack.test\dir 4\test-stack-4.bat" - contains a test and example of using the @callstack macro;
9. ".\tests\wait.mds.tests.events.bat" - full list of event handling tests;
10. ".\tests\wait.events.json" - JSON file used for testing event handling and testing macros too;
11. ".\tests\wait.mds.tests.subroutine.bat" - external procedure testing;
12. ".\tests\wait.mds.samples.installer.bat" - testing the installer;

13. ".\tests\wait.mds.samples.events-mymacro.bat" - an example of converting a simple macro to a string that can be included in a JSON event file;
14. ".\com.api.wrapper\test_batch.bat" - test script that is used by tests;
15. ".\com.api.wrapper\test_forms.exe" - test application used by tests. It is compiled using Microsoft .NET Framework 3.5 SP1 and on later OS versions it will need to be installed locally if it is not installed;
16. ".\com.api.wrapper\- wds.source -\wait.mds.injector.cpp" - the C source file of the injector dynamic library, which is used to read console text. Within the COM server source code, the compiled 32-bit and 64-bit versions of the injector are presented as folded strings.

Project status, support and development.

The source code of the library is functional and can be used on Windows OS versions from XP to 11. The current set of library functionality was formed over several years to solve those automation problems that took place personally for me. This functional set is published here for general use as is. To solve other similar automation problems, the set of library capabilities can be expanded or improved. For offers of financial support and project development, I am available at the following contacts:

E-Mail: kopyurff@yahoo.com, kopyurff@rambler.ru
Mobile: 8-921-912-44-10