Назначение и сигнатуры методов COM-сервера.

Документация библиотеки wait.mds, v2.00.

Библиотека посвящается памяти
Николая и Раисы Копьёвых.

Введение.

Библиотека wait.mds, к которой относится данное описание, является библиотекой поддержки автоматизации на основе MS-DOS скриптов. Приведенное здесь описание посвящено COM-серверу, который используется библиотекой для выполнения действий, необходимых для её функциональности. Набор методов COM-сервера, их функциональность и параметризация определяются их использованием в данной библиотеке. Исходный код сервера написан на VB.NET, поэтому описание сигнатур его методов приводится на этом языке.

В зависимости от версий ОС Windows различаются версии фреймворков .NET на каждом компьютере и также возможно наличие других версий фреймворка на компьютере по усмотрению пользователя. По этой причине откомпилированный файл COM-сервера не включен в библиотеку и поставляется в виде сжатых и свернутых исходных кодов с целью их развертывания, компиляции и регистрации на компьютере пользователя с использованием фреймворка, который установлен локально. Установка COM-сервера осуществляется запуском инсталлятора, который включен в исходный файл библиотеки "wait.mds.bat" и вызывается из командной строки следующим образом:

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

либо путем полной установки библиотеки:

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

Для корректной работы инсталлятора рекомендуется запускать установку под пользователем с правами администратора на локальном компьютере. Если система безопасности ОС не настроена для работы консольных приложений, инсталлятору потребуется произвести перезагрузку компьютера для вступления в силу изменения настроек, после чего установку нужно будет запустить повторно. В зависимости от разрядности ОС, инсталлятор автоматически устанавливает 32-х или 64-х разрядную версию библиотеки. После установки в папке %ProgramFiles% будет создан подкаталог "wait.mds", который будет содержать откомпилированную и зарегистрированную библиотеку COM-сервера, а также его исходный код и вспомогательный файл на VBScript. Содержимое последнего файла также включается в исходный код COM-сервера на VB.NET вместе с соответствующими методами. Рекомендуется для удобства использовать полную установку библиотеки, так как в данном случае регистрируется переменная среды %wait.mds% для запуска основной библиотеки из любой папки, а также создается веб-файл её справки с ярлыком на рабочем столе. Веб-файл справки на английском языке может быть полезен при работе с COM-сервером.

Исходный код COM-сервера работоспособен и может использоваться на ОС Windows от версии XP до версии 11. Windows XP не имеет предустановленного фреймворка .NET, поэтому на ней предварительно должен быть установлен Microsoft .NET Framework 3.5 SP1.

После компиляции и регистрации COM-сервер имеет общедоступное название "WaitMdsApiWrapper". Его использование в исходном коде не отличается от других библиотек этого типа. Пример создания объекта и использования одного из методов на VBScript:

```
Dim wmo
Set wmo = CreateObject("WaitMdsApiWrapper")
wmo.ScreenShot(0, 0, "C:\snapshot.jpg", True)
```

Приведенный выше скрипт делает снимок экрана и сохраняет его в файл "snapshot.jpg" на системном диске.

Список методов и их описание.

Метод ActiveWindow()

Возвращает числовое значение идентификатора текущего активного окна в виде строки.

Синтаксис:

```
ActiveWindow() As String
```

Результат:

Идентификатор текущего активного окна в виде строки.

Параметры:

нет

Метод ForegroundWindow()

Возвращает числовое значение идентификатора окна переднего плана в виде строки.

Синтаксис:

```
ForegroundWindow() As String
```

Результат:

Идентификатор окна переднего плана в виде строки.

Параметры:

нет

Метод WindowOfPoint()

Возвращает значение идентификатора окна, которое содержит указанные координаты.

Синтаксис:

```
WindowOfPoint(ByVal X As Long, ByVal Y As Long) As String
```

Результат:

Идентификатор окна в виде строки.

Параметры:

X — абсолютная X-координата точки;

Y — абсолютная Y-координата точки.

Метод SetForeground()

Делает окно с указанным идентификатором окном переднего плана.

Синтаксис:

```
SetForeground(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

Метод IsWindow()

Осуществляет проверку, является ли числовое значение идентификатором существующего окна.

Синтаксис:

```
IsWindow(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsEnabled()

Осуществляет проверку того, что окно не заблокировано.

Синтаксис:

```
WindowIsEnabled(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае не заблокированного окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsVisible()

Осуществляет проверку того, что окно находится в видимом состоянии.

Синтаксис:

```
WindowIsVisible(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае видимого окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsIconic()

Осуществляет проверку того, что окно находится в свернутом состоянии.

Синтаксис:

```
WindowIsIconic(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае свернутого окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsZoomed()

Осуществляет проверку того, что окно находится в развернутом на весь экран состоянии.

Синтаксис:

```
WindowIsZoomed(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае развернутого окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsChild()

Осуществляет проверку того, что окно является дочерним окном другого окна.

Синтаксис:

```
WindowIsChild(ByVal hWndParent As Long, ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWndParent — идентификатор родительского окна;

hWnd — идентификатор дочернего окна.

Метод WindowIsMenu()

Осуществляет проверку того, что окно является элементом меню.

Синтаксис:

```
WindowIsMenu(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

Метод WindowIsHung()

Осуществляет проверку того, что процесс окна потерял работоспособность.

Синтаксис:

```
WindowIsHung(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае отсутствия отклика окна на тестовые сообщения в течение 5 секунд, то есть при нерабочем состоянии окна.

Параметры:

hWnd — идентификатор окна.

Примечание:

Не применим к окнам консоли, потому как они не имеют обработки очереди сообщений.

Метод ClientRect()

Получает абсолютные координаты клиентской области окна.

Синтаксис:

```
ClientRect(ByVal hWnd As Long, ByRef ALeft As Long, ByRef ATop As Long, ByRef ARight As Long, ByRef ABottom As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна;
ALeft — левая граница клиентской области;
ATop — верхняя граница клиентской области;
ARight — правая граница клиентской области;
ABottom — нижняя граница клиентской области.

Метод WindowRect()

Получает границы окна.

Синтаксис:

```
WindowRect(ByVal hWnd As Long, ByRef ALeft As Long, ByRef ATop As Long,
ByRef ARight As Long, ByRef ABottom As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна;
ALeft — левая граница;
ATop — верхняя граница;
ARight — правая граница;
ABottom — нижняя граница.

Метод WindowClass()

Возвращает класс окна.

Синтаксис:

```
WindowClass(ByVal hWnd As Long) As String
```

Результат:

Строковое имя класса.

Параметры:

hWnd — идентификатор окна.

Метод WindowCaption()

Возвращает заголовок окна.

Синтаксис:

```
WindowCaption(ByVal hWnd As Long) As String
```

Результат:

Строка заголовка окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowText()

Возвращает текст окна.

Синтаксис:

```
WindowText(ByVal hWnd As Long) As String
```

Результат:

Текст окна.

Параметры:

hWnd — идентификатор окна.

Метод WindowInfo()

Возвращает строку с параметрами окна в JSON-формате.

Синтаксис:

```
WindowInfo(ByVal hWnd As Long) As String
```

Результат:

Параметры окна в JSON-формате.

Параметры:

hWnd — идентификатор окна.

Метод ShowWindow()

Изменяет параметры видимости окна в соответствии с переданной командой.

Синтаксис:

```
ShowWindow(ByVal hWnd As Long, ByVal nCmdShow As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

nCmdShow — идентификатор команды для изменения видимости окна. Цифровые значения команды соответствуют значениям команды у Windows API функции ShowWindow, они перечислены в MSDN Microsoft.

Метод MinimizeWindow()

Минимизирует указанное окно.

Синтаксис:

```
MinimizeWindow(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

Метод CloseWindow()

Закрывает указанное окно.

Синтаксис:

```
CloseWindow(ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна.

Метод MoveWindow()

Перемещает указанное окно.

Синтаксис:

```
MoveWindow(ByVal hWnd As Long, ByVal X As Long, ByVal Y As Long, ByVal
nWidth As Long, ByVal nHeight As Long, ByVal bRepaint As Boolean) As Boolean
```

Результат:

True в случае успеха.

Параметры:

hWnd — идентификатор окна;
X — новая X-координата окна;
Y — новая Y-координата окна;
nWidth — новая ширина окна;
nHeight — новая высота окна;
bRepaint — True, если требуется перерисовать окно после перемещения.

Метод SendMessage()

Посылает пользовательское сообщение указанному окну.

Синтаксис:

```
SendMessage(ByVal hWnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

Результат:

Результат соответствует результату Windows API функции SendMessage, которая описана в MSDN Microsoft.

Параметры:

hWnd — идентификатор окна;
wMsg — цифровой идентификатор сообщения;
wParam — первый параметр сообщения;
lParam — второй параметр сообщения.

Метод FindWindow()

Находит окно по его классу и заголовку.

Синтаксис:

```
FindWindow(ByVal lpClassName As String, ByVal lpWindowName As String) As String
```

Результат:

Идентификатор окна в случае успеха, "0" при неудаче.

Параметры:

lpClassName — класс окна;
lpWindowName — заголовок окна.

Метод FindWindows()

Находит окно по его классу и заголовку.

Синтаксис:

```
FindWindows(ByVal lpClassName As String, ByVal lpWindowName As String, ByVal IgnoreCase As Boolean) As String
```

Результат:

Идентификатор окна в случае успеха или найденные идентификаторы через запятую, пустую строку при неудаче.

Параметры:

lpClassName — класс окна;

lpWindowName — заголовок окна;

IgnoreCase — игнорировать регистр букв при поиске.

Примечание:

Параметры "lpClassName" и "lpWindowName" метода допускают использование подстановочных знаков "*" для указания поиска только по подстрокам при любых частях строки ("*"), которые можно игнорировать при поиске.

Метод Repaint()

Посылает команду перерисовать указанное окно.

Синтаксис:

```
Repaint(ByVal hWnd As Long)
```

Параметры:

hWnd — идентификатор окна.

Метод GetTaskInfo()

Получает параметры процесса с указанным PID.

Синтаксис:

```
GetTaskInfo(ByRef pid As Long, Optional ByRef arc As String = "", Optional ByRef app As String = "", Optional ByRef jmpPid As String = "", Optional ByRef wndPid As Long = 0, Optional ByRef wndArc As String = "", Optional ByRef wndApp As String = "", Optional ByVal oldJmpPid As String = "") As String
```

Результат:

Возвращает строку с идентификатором окна, к которому принадлежит процесс.

Параметры:

pid — на входе идентификатор целевого процесса для получения параметров, на выходе — тот же идентификатор процесса, либо идентификатор процесса контекста вызова (если указан "oldJmpPid");

Следующие параметры являются необязательными, используются и указываются для возвращения результата:

arc — архитектура процесса;

app — имя модуля программы;

jmpPid — иерархическая последовательность PID процессов от родительского процесса, которому принадлежит окно, до дочернего целевого процесса. Значение возвращается в виде строки PID, разделенных запятыми (CSV формат);

wndPid — идентификатор процесса окна;

wndArc — архитектура процесса окна;

wndApp — имя модуля процесса окна;

Следующий параметр является необязательным, его значение получается в параметре "jmpPid" при предыдущем вызове:

oldJmpPid — иерархическая последовательность PID от родительского процесса окна до дочернего процесса в CSV формате. Данное входное значение используется для определения процесса, который является общим контекстом вызовов данного метода.

Метод CognateProc()

Возвращает ближайший родственный процесс.

Синтаксис:

```
CognateProc(ByVal pid As Long, ByVal moduleName As String, ByVal parentProc As Byte) As String
```

Результат:

Идентификатор родственного процесса.

Параметры:

pid — идентификатор процесса;

moduleName — имя модуля родственного процесса, пустая строка для игнорирования;

parentProc — True для получения PID ближайшего родительского процесса, False для получения дочернего PID.

Примечание:

При непустом значении параметра "moduleName" ближайшим родственным процессом считается первый процесс с модулем, который совпадает с указанной строкой. Поиск осуществляется без учета регистра букв, имя модуля содержит расширение файла.

Метод PidOfWindow()

Получает идентификатор процесса окна.

Синтаксис:

```
PidOfWindow(ByVal hWnd As Long) As Long
```

Результат:

Идентификатор процесса окна, "0" при несуществующем окне.

Параметры:

hWnd — идентификатор окна.

Метод WindowsOfPid()

Получает разделенный запятыми список идентификаторов окон, которые есть у указанного процесса.

Синтаксис:

```
WindowsOfPid(ByVal pid As Long) As String
```

Результат:

CSV список идентификаторов окон, либо пустая строка.

Параметры:

pid — идентификатор процесса.

Метод FindChildWindows()

Получает разделенный запятыми список дочерних окон, которые есть у указанного окна.

Синтаксис:

```
FindChildWindows(ByVal hWnd As Long, ByVal lpClassName As String, ByVal
lpWindowName As String, ByVal IgnoreCase As Boolean) As String
```

Результат:

CSV список идентификаторов окон, либо пустая строка.

Параметры:

hWnd — идентификатор окна.

lpClassName — класс окна;

lpWindowName — заголовок окна;

IgnoreCase — игнорировать регистр букв при поиске.

Примечание:

Параметры "lpClassName" и "lpWindowName" метода допускают использование подстановочных знаков "*" для указания поиска только по подстрокам при любых частях строки ("*"), которые можно игнорировать при поиске.

Метод MonitorInfo()

Получает информацию по монитору, используя его идентификатор.

Синтаксис:

```
MonitorInfo(ByVal hMonitor As Long) As String
```

Результат:

Получает строку в формате JSON, которая содержит значения атрибутов "cbSize" и "dwFlags" структуры MONITORINFO (см. MSDN).

Параметры:

hMonitor — идентификатор монитора.

Метод MonitorFromPoint()

Получает идентификатор монитора, который содержит точку с координатами.

Синтаксис:

```
MonitorFromPoint(ByVal x As Long, ByVal y As Long) As Long
```

Результат:

Идентификатор монитора.

Параметры:

x — X- координата точки;

y — Y- координата точки.

Метод MonitorFromRect()

Получает идентификатор монитора, который содержит прямоугольник.

Синтаксис:

```
MonitorFromRect(ByVal Left As Long, ByVal Top As Long, ByVal Right As Long,
ByVal Bottom As Long) As Long
```

Результат:

Идентификатор монитора.

Параметры:

Left — X- координата левой грани;

Top — Y- координата верхней грани;

Right — X- координата правой грани;

Bottom — Y- координата нижней грани.

Метод MonitorFromWindow()

Получает идентификатор монитора, который содержит окно.

Синтаксис:

```
MonitorFromWindow(ByVal hWnd As Long) As Long
```

Результат:

Идентификатор монитора.

Параметры:

hWnd — идентификатор окна.

Метод GetScreenResolution()

Возвращает текущее разрешение монитора в параметры.

Синтаксис:

```
GetScreenResolution(ByVal monId As Long, ByRef nWidth As Long, ByRef nHeight As Long)
```

Параметры:

monId — идентификатор монитора;

nWidth — возвращает ширину монитора;

nHeight — возвращает высоту монитора.

Метод NewScreenResolution()

Изменяет разрешение монитора с идентификатором на указанное в параметрах.

Синтаксис:

```
NewScreenResolution(ByVal monId As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Boolean
```

Результат:

True в случае успешного изменения разрешения экрана.

Параметры:

monId — идентификатор монитора;

nWidth — ширина монитора для установки;

nHeight — высота монитора для установки.

Предупреждение:

Передаваемые значения ширины и высоты не могут быть произвольными и должны поддерживаться видеокартой.

Метод ScreenShot()

Получает снимок экрана и сохраняет его в графический файл формата JPG на диске.

Синтаксис:

```
ScreenShot(ByVal monId As Long, ByVal hWnd As Long, ByVal AFullFileName As String, ByVal AClientArea As Boolean)
```

Параметры:

monId — идентификатор монитора, "0" для монитора, содержащего окно или для основного монитора;

hWnd — идентификатор окна или "0". Если нулевое значение, то делает снимок экрана;

AFullFileName — полный путь и имя файла для сохранения снимка;

AClientArea — сделать снимок только клиентской области окна, является значимым только при ненулевом "hWnd".

Примечание:

Если указано сделать снимок окна в свернутом состоянии, то метод его разворачивает для снимка и потом сворачивает к исходному состоянию.

Метод AppBarRect()

Определяет координаты панели задач ОС и возвращает их в параметры.

Синтаксис:

```
AppBarRect(ByRef ALeft As Long, ByRef ATop As Long, ByRef ARight As Long, ByRef ABottom As Long) As Boolean
```

Результат:

True в случае успешного выполнения.

Параметры:

ALeft — X- координата левой грани;

ATop — Y- координата верхней грани;

ARight — X- координата правой грани;

ABottom — Y- координата нижней грани.

Метод ScreenClientArea()

Определяет координаты клиентской области монитора и проверяет, находится ли окно в его пределах.

Синтаксис:

```
ScreenClientArea(ByVal monId As Long, ByVal hWnd As Long, ByRef ALeft As Long,
ByRef ATop As Long, ByRef ARight As Long, ByRef ABottom As Long) As Long
```

Результат:

Возвращает процент поверхности окна в пределах клиентской области монитора или 100 при нулевом идентификаторе окна.

Параметры:

monId — идентификатор монитора;

hWnd — идентификатор окна или нулевое значение;

Следующие параметры служат для получения координат клиентской области монитора:

ALeft — X- координата левой грани;

ATop — Y- координата верхней грани;

ARight — X- координата правой грани;

ABottom — Y- координата нижней грани.

Метод MoveToScrClient()

Перемещает окно в клиентскую область монитора.

Синтаксис:

```
MoveToScrClient(ByVal monId As Long, ByVal hWnd As Long) As Boolean
```

Результат:

True в случае успешного перемещения.

Параметры:

monId — идентификатор монитора;

hWnd — идентификатор окна.

Метод GetCursorPos()

Получает координаты курсора мыши.

Синтаксис:

```
GetCursorPos(ByRef X As Long, ByRef Y As Long) As Boolean
```

Результат:

True в случае успешного выполнения.

Параметры:

X — X- координата точки;

Y — Y- координата точки.

Метод SetCursorPos()

Перемещает курсор мыши в точку с координатами.

Синтаксис:

```
SetCursorPos(ByRef X As Long, ByRef Y As Long) As Boolean
```

Результат:

True в случае успешного выполнения.

Параметры:

X — X- координата точки;

Y — Y- координата точки.

Метод MouseMoveClick()

Выполняет клик мыши, если указано в параметрах предварительно перемещает курсор в точку с координатами.

Синтаксис:

```
MouseMoveClick(ByVal nButton As Long, ByVal bDown As Boolean, ByVal bUp As
Boolean, ByVal bMov As Boolean, ByVal AbsX As Long, ByVal AbsY As Long)
```

Параметры:

nButton — номер кнопки мыши: "1" — левая кнопка, "2" — средняя кнопка и "3" — правая кнопка;

bDown — при клике выполнить нажатие кнопки мыши (True);

bUp — при клике выполнить отпускание кнопки мыши (True);

bMov — предварительно переместить курсор мыши;

AbsX — X- координата нового положения курсора;

AbsY — Y- координата нового положения курсора.

Метод MouseClick()

Выполняет клик мыши.

Синтаксис:

```
MouseClick(ByVal nButton As Long, ByVal bDown As Boolean, ByVal bUp As Boolean)
```

Параметры:

nButton — номер кнопки мыши: "1" — левая кнопка, "2" — средняя кнопка и "3" — правая кнопка;

bDown — при клике выполнить нажатие кнопки мыши (True);

bUp — при клике выполнить отпускание кнопки мыши (True).

Метод CompareFileBytes()

Побайтово сравнивает данные двух файлов.

Синтаксис:

```
CompareFileBytes(ByVal AFilePathA As String, ByVal AFilePathB As String) As Byte
```

Результат:

Возвращает численный результат сравнения, значения могут быть следующими:

1. "0" — данные файлов совпадают;
2. "1" — файлы имеют разные данные;
3. "2" — размер одного из файлов изменился за время их чтения или произошла ошибка чтения;
4. "3" — файлы имеют разные размеры;
5. "4" — не удалось прочитать один из файлов;
6. "5" — один из файлов не был найден.

Параметры:

AFilePathA — имя и расположение первого файла сравнения;

AFilePathB — имя и расположение второго файла сравнения.

Метод GetConsoleText()

Читает текст консоли и возвращает его в виде строки.

Синтаксис:

```
GetConsoleText(ByVal Pid As Long, ByVal lineNo As Long, ByVal lineCount As Long, ByVal lineDelim As String, ByRef AResult As String, Optional ByRef pauseProc As Byte = 7, Optional ByVal AftStr As String = "", Optional ByVal AftSubStr As Boolean = False, Optional ByVal leftTrim As Boolean = False, Optional ByVal widthCWin10 As Integer = 0) As Byte
```

Результат:

Возвращает численный результат чтения текста, значения могут быть следующими:

1. "0" — чтение выполнено успешно;
2. "1" — процесс с PID не имеет окна или не существует;
3. "2" — внутренняя ошибка при выполнении или слишком медленно и остановлено по таймауту;
4. "3" — неожиданные сообщения от инжектора в целевом процессе;
5. "4" — не удалось инжектировать библиотеку, создать наложенное окно или процесс имеет несколько окон;
6. "5" — текстовый буфер консоли был заблокирован;
7. "6" — не удалось получить идентификатор консоли;
8. "7" — процесс не является консольным приложением;
9. "8" — неизвестная архитектура целевого процесса, поддерживаются x86 или x64;
10. "9" — не удалось прочитать или найти файл инжектора на диске, проверьте права доступа;
11. "10" — файл инжектора был успешно создан на диске, вызовите снова для чтения результата;
12. "11" — указанная строка в тексте консоли не была найдена.

Параметры:

Pid — идентификатор процесса окна консоли. При запуске из консольного приложения значение "0" указывает на чтение текста ее консоли;

lineNo — номер строки текста, который следует прочитать. Первым номером строки считается строка над кареткой ввода в консоли;

lineCount — число строк, которое следует вернуть;

lineDelim — разделитель строк, который следует использовать при сворачивании нескольких строк в одну строку результата;

AResult — строка результата, в которую возвращается прочитанный текст. В случае ошибки при выполнении в нее возвращается сообщение ошибки;

Следующие параметры являются необязательными:

pauseProc — параметр для указания способов попыток чтения текста, которые следует использовать

1. "1" — не приостанавливать основной поток окна;
2. "2" — приостанавливать основной поток окна для разблокировки текстового буфера и исключения артефактов чтения из-за текущей печати в окно консоли;
3. "3" — произвести чтение через буфер обмена копированием.

Суммирование параметров 1—3 указывает на комбинирование этих способов, значение по умолчанию "7". Если этот параметр содержит переменную, то при входном значении с "2" в неё будет возвращено значение "1" в случае успешной приостановки потока, "0" — в случае неудачи этого действия;

AftStr — если не пустая строка, то делает незначимым параметр "lineNo" и указывает искать текст после последнего вхождения указанной строки. Поиск совпадения осуществляется без учета регистра букв в строках;

AftSubStr — является значимым только при непустом параметре "AftStr" и указывает искать не точное совпадение, а подстроку в строках консоли;

leftTrim — если True, то в возвращаемых строках текста отбрасываются левые пробелы;

widthCWin10 — параметр применим только к новой консоли, которая появилась начиная с ОС Windows 10. Указывает, какую ширину имеет текст в консоли, может изменяться от "20" до "8192". Значение по умолчанию "80" является стандартной шириной строк текста в старой версии консоли. Если содержит "0", то считается, что длина строк в консоли имеет фактическое значение по текущей ширине окна. Последнее значение работает только при дефолтной ширине шрифтов в консоли, равной 8 пикселям.

Примечание:

Особенностью новой консоли, которая появилась с ОС Windows 10, является удаление символов перевода строк, если при печати длина строки оказалось равной текущей ширине окна консоли. То есть, в данном случае происходит слияние строки со следующей строкой. Поэтому при работе с этой консолью рекомендуется устанавливать её ширину из расчета на длину строк 120 символов, а вывод строк в консоль осуществлять из расчета на стандартную ширину строк 80 символов. Ширина консоли в 120 символов соответствует размеру её окна по умолчанию.

Метод EnvironSet()

Читает переменные окружения, определенные в консольном процессе, и возвращает их в виде строки.

Синтаксис:

```
EnvironSet(ByRef pid As Long, ByRef foundVals As String, Optional ByVal schName
As String = "*", Optional ByVal selMode As Byte = 1) As Byte
```

Результат:

Возвращает численный результат чтения переменных, значения могут быть следующими:

1. "0" — чтение переменных окружения прошло успешно;
2. "1" — запросы Wow64 процессов не поддерживаются (x64);
3. "2" — процесс не является консольным приложением или не числится в списке процессов;
4. "3" — не удалось найти родственный процесс с модулем "cmd.exe";
5. "4" — неизвестная архитектура целевого процесса, поддерживаются x86 или x64;

6. "5" — не удалось открыть целевой процесс;
7. "6" — не удалось запросить базовую информацию процесса;
8. "7" — не удалось прочитать PEB процесса;
9. "8" — не удалось прочитать данные переменных окружения по адресу;
10. "9" — другая ошибка выполнения, с идентификатором ошибки ОС Windows;

Параметры:

Pid — идентификатор консольного процесса. При запуске из консольного приложения значение "0" указывает на чтение переменных одного из процессов текущей консоли;

foundVals — переменные среды, которые были прочитаны в целевом процессе и удовлетворили условиям поиска. Если было обнаружено несколько переменных, то они возвращаются одной строкой с символом перевода строки (16-теричный код символа 0x10) в качестве разделителя;

Следующие параметры являются необязательными:

schName — строка поиска имен переменных, поддерживается использование подстановочных знаков "*" для указания поиска только по подстрокам при частях строки ("*"), которые можно игнорировать при поиске. Если указан только "*", то будут возвращены все переменные с их значениями;

selMode — числовой параметр для указания способа поиска в иерархии процессов консоли:

1. "0" — искать только в процессе с указанным PID;
2. "1" —искать самое позднее определение переменных в последнем дочернем процессе, значение по умолчанию;
3. "2" — искать самое раннее определение переменных от родительского процесса окна консоли.

Метод BatchCallStack()

Получает стек вызова файлов и меток в процессе, выполненных с помощью команды "call" ms-dos.

Синтаксис:

```
BatchCallStack(ByVal cmdPid As Long, Optional ByVal AllBatLab As Byte = 0,
Optional ByRef jobReps As String = "", Optional ByRef tabRes As String = "", Optional
ByRef resCode As Integer = 0, Optional ByVal toDosStr As Boolean = False) As String
```

Результат:

Возвращает строку стека вызова, используя CR символ (0x10) в качестве разделителя между элементами стека, первым в возвращаемом результате идет первый в стеке вызовов элемент. Каждый элемент стека вызова содержит полный список параметров с развернутыми значениями переменных. Элементы стека вызова файлов содержат полный локальный путь до каждого файла, сетевые расположения вызываемых файлов не поддерживаются. В случае ошибки, возвращает "FAIL" и сообщение ошибки в параметр "jobReps".

Параметры:

cmdPid — идентификатор процесса окна консоли, в котором планируется получить стек вызовов;

Следующие параметры являются необязательными:

AllBatLab — числовой параметр для указания способа отбора элементов:
1. "0" — вернуть полный стек вызова пакетных файлов и меток;
2. "1" — вернуть стек вызовов только пакетных файлов;
3. "2" — вернуть стек вызовов только меток;

jobReps — возвращает сообщения об ошибках, при нормальном выполнении возвращает пустую строку;

tabRes — возвращает табулированный результат в следующем виде:
1. все аргументы каждого элемента стека заключены в кавычки;
2. все аргументы каждого элемента стека разделены между собой символом LF (0x13);
3. элементы стека разделены между собой символом CR (0x10);

resCode — возвращает внутренний код ошибки, если произошла. "0" при нормальном выполнении;

toDosStr — если True, то преобразует строки с управляющими символами ms-dos для их корректного отображения в консоли.

Примечания:

1. Из-за внутренних ограничений поиска, метод имеет следующее ограничение на используемые символы в стеке: полные имена файлов с путями, метки, имена переменных и их значения в стеке вызовов должны иметь только символы в диапазоне ASCII кодов от 0x20 до 0x7E. То есть, это могут быть только английские, цифровые символы или другие символы нижней половины диапазона ASCII кодов (смотрите эту таблицу для справки);
2. При наличии в стеке вызовов рекуррентных вызовов возникают неоднозначности в порядке следования его элементов. Для ознакомления со способом обхода этой проблемы, смотрите веб-справку библиотеки к макросу @callstack.

Метод SetCodesOfBaseSymbols()

Предназначением метода является установка соотнесения текстовых символов и цифр системы счисления указанной разрядности.

Синтаксис:

```
SetCodesOfBaseSymbols(ByRef ABase As Integer, Optional ByRef ASyms As String = "", Optional ByRef ACods As String = "")
```

Параметры:

ABase — численный параметр для указания разрядности системы счисления, если определены последующие параметры, то его входное значение игнорируется и в него возвращается значение разрядности, которое было определено из них. При отрицательном входном значении соотнесение осуществляется путем генерации случайной последовательности символов до указанного числа цифр в системе счисления.

Поддерживаются системы счисления разрядностью от "2" до "214". Для исключения проблем с представлением текстовых символов в разных кодировках не рекомендуется использование разрядности больше "86";

Следующие параметры необязательны:

ASyms — строковый параметр для явного указания соотнесения текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение. На примере 16-теричной системы счисления и стандартного соотнесения текстовых символов цифрам, входная строка этого параметра должна быть "0123456789ABCDEF";

ACods — строковый параметр для явного указания соотнесения ASCII hex-кодов текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение. Указание непустой входной строки делает незначимыми входные значения предыдущих параметров. На примере семеричной системы счисления со стандартным соотнесением цифр символам "0123456", входная строка для этого параметра "303132333435".

Примечание:

Потому как внутренняя инициализация установленного соотнесения осуществляется в статический атрибут, то оно сохраняется все время существования COM-объекта пользователя.

Метод DropCodesOfBaseSymbols()

Сбрасывает соотнесение текстовых символов к цифрам системы счисления, ранее указанное методом SetCodesOfBaseSymbols().

Синтаксис:

```
DropCodesOfBaseSymbols()
```

Параметры:

нет

Метод Radix()

Осуществляет преобразование числа в виде массива цифр из одной системы счисления в другую.

Синтаксис:

```
Radix(ByVal ASrcRad As UInt16, ByVal ATgtRad As UInt16, ByRef ANumber() As Integer, Optional ByVal ATgtRank As Integer = -1)
```

Параметры:

ASrcRad — неотрицательный численный параметр для указания исходной разрядности системы счисления;

ATgtRad — неотрицательный численный параметр для указания целевой разрядности системы счисления;

ANumber — целочисленный массив, на входе должен содержать число в исходной системе счисления, на выходе содержит число в целевой системе счисления. Число в массиве содержится в виде численных значений цифр по разрядам. Старшие разряды числа находятся в первых элементах массива, младшие - в последних;

ATgtRank — численный параметр для указания размера числа после преобразования к целевой системе счисления. Если указан размер меньше его размера, то отбрасываются старшие разряды, если больше - старшие разряды дополняются нулями до указанного размера. При любом отрицательном значении в этом параметре будут отброшены незначащие нули старших разрядов, значение по умолчанию "-1".

Метод StrRadix()

Осуществляет преобразование строкового представления числа из одной системы счисления в другую.

Синтаксис:

```
StrRadix(ByVal ASrcMap As Boolean, ByRef ASrcRad As UInt16, ByRef ASrcSym As String, ByRef ASrcCod As String, ByVal ANumber As String, ByVal ATgtMap As Boolean, ByRef ATgtRad As UInt16, Optional ByRef ATgtSym As String = "", Optional ByRef ATgtCod As String = "", Optional ByVal ATgtRank As Integer = -1) As String
```

Результат:

Возвращает строковое представление числа в целевой системе счисления.

Параметры:

ASrcMap — исходная строка содержит число с соотнесением цифр текстовым символам (True) или шестнадцатеричную последовательность цифр в числе (False);

ASrcRad — неотрицательный численный параметр для указания исходной разрядности системы счисления;

ASrcSym — строковый параметр для явного указания соотнесения текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение (ASrcMap <==> True);

ASrcCod — строковый параметр для явного указания соотнесения кодов текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение (ASrcMap <==> True);

ANumber — исходная строка с числом для конвертирования;

ATgtMap — целевая строка будет содержать число с соотнесением цифр текстовым символам (True) или шестнадцатеричную последовательность цифр в числе (False);

ATgtRad — неотрицательный численный параметр для указания целевой разрядности системы счисления;

Следующие параметры являются необязательными:

ATgtSym — строковый параметр для явного указания соотнесения текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение (ATgtMap <==> True);

ATgtCod — строковый параметр для явного указания соотнесения кодов текстовых символов числам системы счисления, если пустая строка, то в него возвращается сгенерированное соотнесение (ATgtMap <==> True);

ATgtRank — численный параметр для указания размера числа после преобразования к целевой системе счисления.

Примечание:

После выполнения метод восстанавливает старую базу счисления, если она была установлена ранее. Для дополнительной информации по параметрам "ASrcSym", "ASrcCod", "ATgtSym", "ATgtCod" смотрите метод SetCodesOfBaseSymbols(), для дополнительной информации по "ATgtRank" — Radix(). Если в параметрах "ASrcSym" и "ASrcCod" нет необходимости, то вместо них могут быть указаны пустые строки.

Метод Shrink()

Основным предназначением метода является сжатие текста, преобразование результата к строковому виду для его последующего хранения в текстовом виде и обратное преобразование. Для сжатия данных используется алгоритм Лемпеля — Зива — Велча (LZW) с последующим адаптивным арифметическим кодированием. Преобразование к текстовому виду осуществляется путем преобразования сжатого массива байтов к системе счисления с разрядностью, которая позволяет соотнести числа текстовым символам. Изменение параметров позволяет менять его функциональность.

Синтаксис:

```
Shrink(ByVal AShrink As Boolean, ByRef ASrc As String, ByVal AFSrc As Boolean,
ByRef ATgt As String, ByVal AFTgt As Boolean, Optional ByVal AType As Byte = 2,
Optional ByVal AUCase As Boolean = False, Optional ByVal ASplitRank As UInt16 =
16, Optional ByVal AUTF8 As Boolean = False, Optional ByVal AStrRadix As Byte =
84, Optional ByVal AFileRowLen As UInt16 = 5000, Optional ByVal AEcho As
Boolean = False) As Boolean
```

Результат:

Возвращает True в случае успешного выполнения.

Параметры:

AShrink — параметр для указания того, осуществляется сжатие (True) или вызов для распаковки (False);

ASrc — строковый параметр для указания исходного текста при сжатии, либо сжатых данных при распаковке;

AFSrc — указывает на то, что предыдущий параметр содержит не данные, а имя файла с данными (True);

ATgt — строковый параметр для возврата сжатых данных при сжатии, либо исходного текста при распаковке;

AFTgt — указывает на то, что предыдущий параметр будет содержать не данные, а имя файла с данными (True);

Следующие параметры необязательны, по умолчанию имеют оптимальное значение:

AType — числовой параметр для указания типа преобразования, может иметь следующие значения:
- "0" <=> простое преобразование LZW, может применяться только для сжатия в двоичный файл;
- "1" <=> преобразование LZW с арифметическим кодированием для сжатия в двоичный файл;
- "2" <=> значение по умолчанию. Преобразование LZW с кодированием, результат преобразуется в строковое представление чисел в указанной системе счисления, используя выбранное соотнесение цифр к текстовым символам;
- "3" <=> преобразование LZW с кодированием, результат представляет собой шестнадцатеричное строковое представление байтов сжатого массива как есть;

AUCase — указывает на то, что исходный текст может быть весь конвертирован к верхнему регистру (True), по умолчанию False. Является значимым только при вызове метода для сжатия;

ASplitRank — определяет длину байтовых подмассивов в результирующем сжатом массиве, которые будут преобразовываться в строки (числа) указанной системы счисления, по умолчанию "16". Минимальное значение "8", а из-за ограничений производительности преобразования не рекомендуется выбирать значение больше "32";

AUTF8 — указывает на то, что исходный текстовый файл может быть файлом Unicode со служебной последовательностью символов с кодами "&#EF;&#BB;&#BF" в его начале. Является значимым только при сжатии, по умолчанию False;

AStrRadix — разрядность системы счисления для преобразования сжатых данных в текстовый вид, значение по умолчанию "84". Значение "86" является максимальным "надежным" значением с одинаковым представлением символов в разных кодировках. Является значимым только при значении "AType", равному "2";

AFileRowLen — длина строк при записи в файл сжатых данных в виде текста, по умолчанию "5000". Является значимым только при значении "AType", равному "2";

AEcho — указывает на то, что результат нужно напечатать на экран (True), а не возвращать в "ATgt", по умолчанию False. Является значимым только при "AFTgt", равному False.

Примечание:

Если предполагается использовать встроенное соотнесение текстовых символов и цифр системы счисления, то при равном "2" параметре "AType" метод не требует предварительного вызова SetCodesOfBaseSymbols().

Метод GetDosString()

Осуществляет простые преобразования строки для корректного отображения управляющих символов в консоли ms-dos.

Синтаксис:

```
GetDosString(ByVal ASrcString As String, AForEcho As Boolean) As String
```

Результат:

Возвращает строку с необходимыми преобразованиями.

Параметры:

ASrcString — исходная строка;

AForEcho — в случае True напечатает преобразованную строку на экран вместо ее возврата (False).