Virtual Serial Port Driver

SDK User Guide

Introduction

Nowadays COM ports are still frequently used in many areas: in legacy applications (such as industrial automation systems), shop till systems, and lots of other products. But using devices or software requiring a serial connection, you may face such a problem that you don't have any free serial ports left.

Virtual Serial Port Driver comes to aid, creating virtual serial ports which appear in the system like "standard" hardware ones. Now you can work easily with a device, requiring a COM port connection (GPS device, printer, scanner, or some other peripheral) from several serial port applications simultaneously.

Virtual Serial Port Driver combines three Electronic Team products - Serial Splitter, Shared Serial Ports, and COM Port Redirector. It's a great chance for those who want to use different scenarios of port bundle creation, and set custom port parameters. Virtual Serial Port Driver provides you with the advanced possibilities to **manage real and virtual ports** up to your demands:

1. The software makes it possible to provide your system with an unlimited number of virtual COM port pairs. Thanks to the advanced **pairing option**, you can exchange data between serial apps using a virtual pair connected via a virtual null-modem cable.

2. With Virtual Serial Port Driver you can **split** one real port into several virtual ones. Thus data coming into the split real port will be sent to each of the out-side virtual ports, and vice versa.

3. Another great feature is **joining** several real ports into one virtual. With this option, everything sent to any joined port will be duplicated to the joining one, and vice versa.

4. In case you need to create a bundle of real or virtual serial interfaces where all ports will be able to talk to each other, you can easily do this with the apps' **merging feature**. Everything sent to one of the merged ports can be replicated to all the other ports of the bundle.

5. The **port switching** option will let you add several real ports to one port bundle and join the bundle to a single virtual COM port. Once you create a switcher for this virtual serial port in your app, your program gets access to all real ports joined to the virtual one. Now, any time your app needs to connect to a COM port it will be automatically offered a free real interface. The app will connect to different physical interfaces as though it were

always the same port.

6. Also, Virtual Serial Port Driver gives a way to **redirect** serial traffic from one real port to another real or virtual COM port.

7. By using the software, you can **share** one real port among multiple applications at a time. All applications added to a bundle will be able to communicate data with the same serial port simultaneously. Each application will think that it is working with the serial port in exclusive mode. You can control the access rights of each application and specify serial port connection parameters (baudrate, databits, etc).

8. With Virtual Serial Port Driver, it’s easy to create **loopback** connections where serial data will be coming in and out of the same serial port.

9. Besides, for your most productive work, all those amazing options mentioned above are combined in the unique **complex bundles** feature, which lets you create multiple serial ports at both sides of a bundle, with data redirected from several incoming ports to several outgoing ports. Both sides of the bundle can have virtual serial ports, real serial ports, and shared ports installed. Using the technology of setting main ports at both sides of the bundle, you will be able to easily manage hardware control lines.

Virtual Serial Port Driver offers the Standard and Professional (PRO) editions.

The Standard version is designed to create pairs of virtual serial ports linked via a virtual null-modem connection.

The PRO version gives a way to set up and manage advanced bundles of real and virtual serial ports that allow joining, splitting, redirecting, and sharing serial port data.

The following are the key features of the Virtual Serial Port Driver Standard edition:

- Any number of virtual serial port pairs can be created.
- A virtual serial port may have any name you like, even the same name as a real port has.
- Printers can be assigned to any virtual serial port (available after Windows reboot).
- Full emulation of hardware control lines.
- Serial ports are created and re-configured in real-time.
- Full support of a wide range of GPS devices and software.
- Full compatibility with HyperThreaded and multi-processor systems.
- Bidirectional data transfer.
- High-speed data exchange from/to virtual serial ports.
- Full baudrate and flow control support.
- Hot installation without any need to reboot.
- Customization of real serial port connection parameters (baudrate, parity, data bits, and stop bits).

- Possibility to control serial ports directly from your own app by executing the DLL functions.
- Virtual serial ports are absolutely the same copies of real ones – applications won't see the difference between real and virtual serial ports.
- The link between virtual serial ports is much faster than a real null-modem cable connection and solely depends on your processor speed (average transfer speed is about 5.5 Mbytes/sec).
- Availability of sample application code for the major development environments.
- Compatibility with Windows XP/2003/2008, Windows Vista, Windows 7/10/11, Windows Server 2012/2016/2019, support for Windows on ARM-based systems.

Note: the Virtual Serial Port Driver GUI is compatible with Windows versions starting from Windows 7 and later.

The advanced features supported by Virtual Serial Port Driver PRO include:

- Creating pairs of virtual serial ports.
- Splitting a real port into several virtual ones.
- Joining several real serial ports into one virtual.
- Merging multiple real and virtual ports within a single bundle.
- Redirecting dataflows from real ports to other real or virtual ports.
- Sharing one real serial port between several applications.
- Organizing complex bundles with multiple ports added to the input and output sides.
- Joining several real ports into one virtual for their further automatic switching.
- Creating loopback connections using the same serial port for sending and receiving serial data.

Note: The features to create a port switcher, to share access to a real serial port and to add a shared port to a complex port bundle are currently unavailable for ARM-based Windows systems.

When purchasing the SDK License, you receive an SDK activation code for Virtual Serial Port Driver quiet activation on end-user machines. Depending on your license type you are provided with the activation code either for the Standard or PRO edition of the software.

Whenever you need to check the software version activated on your computer, you can start the software UI and select ***Help*** > ***About*** from the main menu.

SDK Redistribution

Drivers installation

1. Depending on the operating system your target machine runs, select one of the following subdirectories and copy it to the target computer:

    a. **NT5** - for Windows XP/2003.
    b. **NT5x64** - for 64-bit Windows XP/2003.
    c. **NT6** - for Windows Vista/7/8/8.1/10/11, Windows Server 2012/2016/2019..
    d. **NT6x64** - for 64-bit Windows Vista/7/8/8.1/10/11, Windows Server 2012/2016/2019.
    e. **NT6ARM64** - for ARM-based Windows systems.

2. Copy the **vspdxp_install.exe** and **ss_install.exe** files to the directory to which you copied the above-mentioned subdirectory with driver files.

3. To start the automatic installation process, execute the **vspdxp_install.exe** and **ss_install.exe** commands.

4. To install Switcher drivers, launch **setup_sersw.exe** from the respective folder.

5. If you plan to provide the end-user with the uninstallation option, keep the **sershare_setup.exe** and **vsbsetup.exe** files on the target system.

6. To start the uninstallation process, execute the **vspdxp_install.exe /u** and **ss_install.exe /u** commands.

The table below shows the driver subdirectory content and its description:

File	Description
VSPD driver installation	
Evsbc9.inf	.INF file needed for correct installation of Virtual Serial Bus Driver
Evsbc9.sys	Virtual Serial Bus Driver

Vsbsetup.exe	Executable that will silently install Virtual Serial Bus Driver

Evserial9.inf	.INF file needed for correct installation of Virtual Serial Port Driver
Evserial9.sys	Virtual Serial Port Driver
Evsbc9.cat	Catalog of files containing the necessary information for drivers signature
Evserial9.cat	Catalog of files containing the necessary information for drivers signature
Shared Serial Ports drivers installation	
Sershare.inf	.INF file needed for correct installation of Shared Serial Port driver
Sershare.sys	Shared Ports Driver
Sershare_setup.exe	Executable that will silently install Shared Serial Bus driver
Spbus.inf	.INF file needed for correct installation of Shared Serial Port driver
Spbus.sys	Shared Serial Bus Driver
Sershare.cat	Catalog of files containing the necessary information for drivers signature
Spbus.cat	Catalog of files containing the necessary information for drivers signature
Switcher drivers installation	
SerSw.cat	Catalog of files containing the necessary information for drivers signature
SerSw.inf	.INF file needed for correct installation of Switcher drivers
sersw.sys	Switcher driver
setup_sersw.exe	Executable that will silently install Switcher driver

Working with Virtual Serial Port Driver Service

Distribution of Virtual Serial Port Driver technology inside your product can be achieved by installing the Virtual Serial Port Driver Service on the end-user systems.

To install the Virtual Serial Port Driver Service in the target environment, do the following:

1) First, on the developer's machine, activate Virtual Serial Port Driver using the SDK activation code you received from us after your SDK license purchase:

a. Launch Virtual Serial Port Driver and select ***Help*** > ***Enter activation code*** from the main menu.

b. In the Activation window, enter your SDK activation code and click the ***Activate*** button.

Once activated, the file **vspdpro.act** will appear in the following directory:

"C:\ProgramData\Electronic Team\VSPDPro\vspdpro.act"

When installing your own software on the end-user computer, you need to copy the **vspdpro.act** file and place it in the same directory as the **vspdpro_service.exe** file and your application installer file.

2) Install the Virtual Serial Port Driver Service on the end-user system by executing the command "vspdpro_service.exe install".

If the Service is installed successfully, you will receive the "OK" message or "Error" message otherwise.

3) Now, you can start the Service by executing the "vspdpro_service.exe enable" command.

From now on, the Service will be automatically started every time the PC it is installed on is rebooted.

Service uninstallation

If you want to uninstall Virtual Serial Port Driver Service, execute the following:

1. If the Service is currently running, you need to stop it using the "**vspdpro_service.exe**

disable" command.

2. Execute the "**vspdpro_service.exe uninstall**" command-line operation to uninstall the Service.

3. You can also uninstall the drivers for virtual and shared ports using the "**vsbsetup.exe /u**" and "**sershare_setup.exe /u**" commands.

Working with Windows Registry

Setting Virtual Serial Port Driver options

All configurations are located in the registry:

for 32-bit system:

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]

for 64-bit system:

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Electronic Team\VspdPro]

Via **[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]**

variables	***description***
SendDataOnlyMainPort=dword:1 or 0	Allow sending data to main port only
IgnoreError=dword:1 or 0	Ignore errors while creating bundles
AutoOpen=dword:1 or 0	Retry opening real port later in case of failure
Prints=dword:1 or 0	Make virtual and shared ports available for printers

Via **[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\evserial]**

variables	***description***
StrictBaudrate=dword:1 or 0	Enable baudrate emulation for virtual serial ports

Each bundle consists of the following variables:

variables	***description***
[Name_bundle]	A key that contains bundle name
type=Type	Bundle type (behavior)

InMain=PortName	A string variable that contains the name of the main port on the "IN" side of the bundle
OutMain= PortName	A string variable that contains the name of the main port on the "OUT" side of the bundle
StrictBaudrate	1 means that baudrate emulation is enabled for virtual ports in the bundle, 0 if baudrate emulation is disabled
DTR=SignalLines	Contains signal lines DTR is connected to. Available only in Pair (type=7) and Loopback(type=8) bundles
RTS=SignalLines	Contains signal lines RTS is connected to. Available only in Pair (type=7) and Loopback(type=8) bundles
User=UserSession	A string that contains user session identifier
[Name_bundle\In]	A key that contains the list of ports on the "IN" side of the bundle
list_ports	String variables that contain the names of all ports on the given bundle side
[Name_bundle\Out]	A key that contains the list of ports on the "OUT" side of the bundle
list_ports	String variables that contain the names of all ports on the given bundle side

Name_bundle – bundle name

Type - Int variable that contains bundle type.

Name_bundle\In – key that contains the list of ports on the IN-side.

Name_bundle\ Out – key that contains the list of ports on the OUT-side.

InMain – string variable the value of which is the main port on the IN-side.

OutMain – string variable the value of which is the main port on the OUT-side.

DTR, RTS - int variables that describe the pinout of signal lines in Pair and Loopback bundles.

User - if not empty, this string variable will contain the identifier of the user session for which the specified port pair has been created.

list_ports – string variables that contain the names of all ports on the given bundle side. They are of the following type: PortName= PortType[|PortParams], where

PortName - port name, for example: COM6

PortType – port type: Virtual, Real, Shared, Switcher, or VSPD port.

“VSPD ports” are virtual ports that are used only in Pair and Loopback port bundles.

PortParams – Parameters to open a port. Only for real ports. Parameters are set as follows: nBaudrate, cParity, nDatabits, nStopBits, cFlowControl.

If these parameters are not set, then real port parameters will depend on the parameters of the main port of the opposite side. To transfer all parameters, the opposite main port should be of Virtual or Share type. If a port is of Real type, it will open with the last applied parameters.

nBaudRate

Defines serial port baudrate. Possible values are: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000 and 256000 bauds. 9600 is selected by default.

nDatabits

Defines data lengths in a packet. Possible values are: 5, 6, 7 and 8. "7" is selected by default.

cParity

Defines parity is one of the following values: "No parity", "Odd", "Even", "Mark" and "Space". Default value is "No parity".

nStopBits

Defines a number of stop bits. Possible values are: " "1 stop bit" and "2 stop bits". The default value is "1 stop bit".

cFlowControl

Defines flow control value. Possible values are: "None", "Hardware", "Xon/Xoff". Default value is "None".

PortSettings parameters could be empty, which means that dynamic connection parameters will be used instead.

Permissions settings for ports of Share type

- **[HKLM\Software\ELTIMA Software\SharedSerialPorts2]**
    - Shared Port Name
        - Permissions
            - List custom permissions for programs

Variable name	***Data***
Full path	A DWORD value containing OR-ed permission flags. By default, the value is Read|Write (0x03) Read - 0x01 Write - 0x02 Control - 0x04

Examples of port bundles creation

- **Complex Bundle**

HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\VspdPro
"type"=4
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\VspdPro\Complex#1

"OutMain"="COM3"
"InMain"="COM1"

HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\VspdPro\Complex#1\IN

"COM1"="Real|"
"COM2"="Real|"

HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\VspdPro\Complex#1\OUT

"COM3"="Virtual"
"COM4"="Virtual"
"COM5"="Shared"
Data comes to the “IN” side of the bundle from COM1 (real port, set as main one) and then is split into several flows and directed to different ports: COM3 (virtual, main), COM4 (virtual), COM5 (shared). And vice versa data comes to the IN side from COM3 (virtual, main), COM4 (virtual) and COM5 (shared) and is joined into a single flow: COM1 (real port, set as main one).

Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
AddBundle("Complex bundle (1)");
BundleAddPort("Complex bundle (1)", portInput, "COM1", portReal);
BundleAddPort("Complex bundle (1)", portInput, "COM2", portReal);
BundleSetMainPortForSide("Complex bundle (1)", portInput, "COM1");
BundleAddPort("Complex bundle (1)", portOutput, "COM3", portVirtual);
BundleAddPort("Complex bundle (1)", portOutput, "COM4", portVirtual);
BundleAddPort("Complex bundle (1)", portOutput, "COM5", portShared);
BundleSetMainPortForSide("Complex bundle (1)", portOutput, "COM3");
```

- **Split**

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\\VspdPro]

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\\VspdPro\Split#1]

"OutMain"="COM2"
"InMain"="COM1"

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\\VspdPro\Split#1\IN]

"COM1 [Communications Port]"="Real|"

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Electronic Team\\VspdPro\Split#1\OUT]

"COM2"="Virtual"
"COM3"="Virtual"

COM1 (real port, set as main one) is split into several virtual ones: COM2 (main) and COM3.
Real port (COM1) parameters are dynamic and depend on the parameters of the main port of the opposite side (COM2).
Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
AddBundle("Split port (1)"); // or AddBundleEx("Split port (1)",
bundleSplit)
BundleAddPort("Split port (1)", portInput, "COM1 [Communications Port]",
portReal);
BundleSetMainPortForSide("Split port (1)", portInput, "COM1");
BundleAddPort("Split port (1)", portOutput, "COM2", portVirtual);
BundleAddPort("Split port (1)", portOutput, "COM3", portVirtual);
BundleSetMainPortForSide("Split port(1)", portOutput, "COM2");
```

- **Join**

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]

"SendDataOnlyMainPort"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Joined ports (2)]

"OutMain"="COM3"
"InMain"="COM2"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Joined ports (2)\IN]

"COM1 [Communications Port]"="Real|4800,O,6,2,X"
"COM2 [Communications Port]"="Real|"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Joined ports
(2)\OUT] "COM3"="Virtual"
```

Several real ports COM1 and COM2 (main) are joined into one virtual port (COM3). Everything sent to the joining port (COM3) will be duplicated to all joined ones: COM1 and COM2 (main). Real port parameters are the following: baudrate - 4800, parity - odd, databits - 6, stopbits - 2, flow control - Xon/Xoff.

Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
OptionSetOnlyMainPortCanSend(True); // Warning: This can effect on other
bundles
AddBundle("Joined ports (2)"); // or AddBundleEx("Joined ports (2)",
bundleJoin)
BundleAddPort("Joined ports (2)", portInput, "COM1 [Communications
Port]", portReal);
BundleSetRealPortParameters("Joined ports (2)", portInput, "COM1
[Communications Port]", 4800, 1, 6, 2, 1);
BundleAddPort("Joined ports (2)", portInput, "COM2 [Communications
Port]", portReal);
BundleSetMainPortForSide("Joined ports (2)", portInput, "COM2");
BundleAddPort("Joined ports (2)", portOutput, "COM3", portVirtual);
```

```
BundleSetMainPortForSide("Joined ports (2)", portOutput, "COM2");
```

- **Redirect**

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Redirect#1]

"OutMain"="COM3"
"InMain"="COM2"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic

Team\VspdPro\Redirect#1\IN] "COM2 [Communications Port]"="Real|"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic

Team\VspdPro\Redirect#1\OUT] "COM3"="Virtual"

All serial traffic from real port (COM2) is redirected to another port (COM3, virtual).Real port parameters are dynamic and depend on the parameters of the main port of the opposite side (COM3).

Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
AddBundle("Redirect#1"); // or AddBundleEx("Redirect#1", bundleRedirect)
BundleAddPort("Redirect#1", portInput, "COM2 [Communications Port]",
portReal);
BundleSetDynamicParametersForRealPort("Redirect#1", portInput, "COM2
[Communications Port]");
BundleSetMainPortForSide("Redirect#1", portInput, "COM2");
BundleAddPort("Redirect#1", portOutput, "COM3", portVirtual);
BundleSetMainPortForSide("Redirect#1", portOutput, "COM2");
```

- **Share**

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Shared port (1)]

"OutMain"="COM1"
"InMain"="COM1"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Shared port

(1)\IN] "COM1"="Real|"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro\Shared port

(1)\OUT] "COM1"="Shared"

The VSPD Service opens the real port COM1 and creates a virtual port that will be shared among multiple applications at a time. All data sent to the virtual port by any of the connected applications will be redirected to the real port by the VSPD Service.

Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
AddBundle("Shared port (1)"); // or AddBundleEx("Shared port (1)",
bundleShare) BundleAddPort("Shared port (1)", portInput, "COM1",
portReal);
BundleSetMainPortForSide("Shared port (1)", portInput, "COM1");
BundleAddPort("Shared port (1)", portOutput, "COM3", portShared);
BundleSetMainPortForSide("Shared port (1)", portOutput, "COM2");
```

- **Pair**

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic Team\VspdPro]

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic
Team\VspdPro\Pair#3] "DTR"=0xa0
"RTS"=0x10
"User"="00000000-0504fed5"
"OutMain"="COM3"
"InMain"="COM1"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic
Team\VspdPro\Pair#3\IN] "COM1"="VSPD||1|1"

[HKEY_LOCAL_MACHINE\SOFTWARE\Electronic
Team\VspdPro\Pair#3\OUT]
"COM3"="VSPD||1|1|-c:\windows\system32\cmd.exe"

The paired virtual ports COM1 and COM2 are created for the user with session identifier "00000000-0504fed5". Cmd.exe will not be able to open the serial port COM3. The pair will have the DTR line connected to DSR and DCD, and the RTS line connected to CTS.

Below is an example of how to create a port bundle of this type (in pseudocode without error checking):

```
AddBundleEx("Pair#3", bundlePair);
PairSetUserSession("Pair#3", "00000000-0504fed5");
BundleAddPort("Pair#3", portInput, "COM1", portVirtual); // DLL hides
VSDP/Virtual ports difference
BundleAddPort("Pair#3", portOutput, "COM3", portVirtual);
String[] accessList = { "-c:\windows\system32\cmd.exe"};
VirtualSetAccessList("Pair#3", portOutput, "COM3", accessList);
```

Command-Line Options of Virtual Serial Port Driver

Virtual Serial Port Driver lets you use command-line options to automate the process of creating and managing bundles of real and virtual serial ports.

A **port bundle** is a collection of real and/or virtual serial ports divided by data transfer direction.

Virtual Serial Port Driver supports the following bundle types:

Split, pair, join, merge, switch, redirect, share, loopback, complex.

Note: Different bundle types may affect the behavior of certain port types.

The default bundle type is *Complex*.

Port names:

Port names may contain A-Z, a-z, 0-9, and the underline character (_). For example, "COM1", "COM_8".

Port types:

Depending on the bundle type, you can use serial ports of the following types:

- Real (physical) ports.
- Virtual ports.
- Shared ports.
- Switcher ports.
- vspd, paired, pair, p (only for pair and loopback port bundles)

The default port type is *virtual*.

Bundle names:

Port bundle names may contain Latin letters, digits, and special characters. For example, "Pair#1", "Split#1".

Directions:

There are two data transfer directions in port bundles created using Virtual Serial Port Driver: *Input* and *Output*.

A shared port:

You can create a port bundle where one real port will be shared and used by multiple applications at the same time.

1. *Help* commands

To invoke help for a certain command, use the following:

HELP ["command name"|all|bundles|ports|names|direction|share] …

> ***examples:***
>
> *vspdpro_service.exe help install*
>
> *vspdpro_service.exe help uninstall*
>
> *vspdpro_service.exe help all*
> *\\Display general help.*

To get a short description of how to use the command-line interface, execute:

USAGE ["command name"] ...

2. Commands to manage the Virtual Serial Port Driver Service and register the software.

INSTALL
Installs the VSPD Service.

> ***example:***
> *vspdpro_service.exe install*

UNINSTALL
Uninstalls the VSPD Service.

> ***example:***
> *vspdpro_service.exe uninstall*

ENABLE
Enables the VSPD Service.

> ***example:***
> *vspdpro_service.exe enable*

DISABLE
Disables the VSPD Service.

> ***example:***
> *vspdpro_service.exe disable*

REG "key"
Used to register your copy of Virtual Serial Port Driver.

> ***example:***
> *vspdpro_service.exe REG XXXXX-12345-12345-12345-XXXXX*

AUTOSTART
Enables the VSPD Service to run at boot.

> ***example:***
> *vspdpro_service.exe autostart*

3. Commands to create and manage serial port bundles.

LIST
Displays the list of all available port bundles.

> ***example:***
> *vspdpro_service.exe list*

SHOW ["bundle"] ...
Shows detailed info about the selected bundle.

> ***example:***
> *vspdpro_service.exe show Pair#1*

SET-CONFIG
Changes the Service settings.

> **[/ignore-errors yes|no]**
> **[/baudrate-emulation   yes|no]**

[/printers yes|no]
[/reopen yes|no]

ignore-errors
Ignores errors when creating bundles

With this option enabled, you can continue the creation/opening process even if some of the ports are not created/opened.

baudrate-emulation
Enables strict baudrate emulation.

Strict baudrate emulation will be enabled by default when creating bundles with virtual serial ports.

printers
Makes ports available for printers.

You can connect a printer to any created virtual port. This is achieved by registering virtual serial ports in the Windows Registry so that they can be selected in the Add Printer Wizard.

After enabling this option, you'll see the warning message about the need to restart the "Print Spooler" service or reboot your computer for this feature to come into operation.

reopen
Retries opening real ports in case of failure.

This option allows reopening serial ports automatically if they failed to open on the first go (and the "Ignore errors…" option is enabled) or a serial port was removed while the Service was still running (e.g. the USB to COM lead was unplugged).

> ***Example:***
> *vspdpro_service.exe set-config /ignore-errors yes /baudrate-emulation yes /printers yes /reopen yes*

ADD-BUNDLE "name" ["bundle-type"]
Creates a new port bundle.

Supported bundle types (and their shortcuts):

split, s
join, j
redirect, redir, r
share, sh
complex, c, advanced, adv, a

switcher, switch, sw
merge, m
vspd, pair, p
loopback, loop, l

Example:
vspdpro_service.exe add-bundle Pair#1 p

RENAME-BUNDLE "old name" "new name"
Renames a port bundle.

Example:
vspdpro_service.exe rename-bundle Pair#1 Pair#2

DELETE-BUNDLE "bundle"
Deletes a port bundle.

Example:
vspdpro_service.exe delete-bundle Pair#1

ADD-PORT "port" "bundle" [in|out] ["port-type"]
Adds a new port to a bundle.

Note: Before using this command, make sure that the bundle you would like to add the port to exists.

Supported port types (and their shortcuts):

- real, r
- virtual, v
- shared, share, sh
- switcher, switch, sw
- vspd, paired, pair, p (only for pair and loopback port bundles)

Examples:
vspdpro_service.exe add-port COM1 Pair#1 in virtual
vspdpro_service.exe add-port COM2 Pair#1 out virtual

\\Add the virtual port COM1 to the Input side and the virtual port COM2 to the Output side in the bundle Pair#1

SET-PORT "port" "bundle" [in|out]
Sets custom settings for a real port (e.g. baudrate, parity, etc.)

[/baudrate   rate]

```
[/read  yes|no]
[/write yes|no]
[/parity        none|odd|even|mark|space]
[/databits      5..8]
[/stopbits      1..2]
[/flow  none|hw|x]
```

Example:
vspdpro_service.exe set-port COM1 Split#1 in /baudrate 300 /read yes /write yes /parity even /databits 7 /stopbits 1 /flow hw

\\Configure settings for the real port COM1 added to the Input side of the port bundle Split#1: baudrate=300 read=yes write=yes parity=even databits=7 stopbits=1 flow=hw

UNSET-PORT "port" "bundle" [in|out]
Resets custom settings for a real port.

example:
vspdpro_service.exe unset-port COM1 Split#1 in

DELETE-PORT "port" "bundle" [in|out]
Removes port from a bundle.

example:
vspdpro_service.exe delete-port COM5 Split#1 out

SET-MAIN "port" "bundle" [in|out]
Sets port as main at the given direction.

example:
vspdpro_service.exe set-main COM5 Split#1 out

UNSET-MAIN "bundle" [in|out]
Unsets all main ports on the selected bundle side (In or Out).

example:
vspdpro_service.exe unset-main Split#1 out

SET-SHARE "port" "bundle" [in|out] "app name" [rwc|0..7]
Adds an application to the list of applications that are able to access the shared port.

Note: You should specify the port type “shared” when using this command.

[rwc|0..7] - access rights:

“read” - allows reading port data.
“write” - allows writing data to the port.
“control” - allows controlling port settings.

0 - *Read/Write/Control* off.
1 - *Read* on, *Write/Control* off.
2 - *Write* on, *Read/Control* off.
3 - *Read/Write* on, *Control* off.
4 - *Control* on, *Read/Write* off.
5 - *Read/Control* on, *Write* off.
6 - *Write/Control* on, *Read* off.
7 - *Read/Write/Control* on.

> ***Example:***
> *vspdpro_service.exe set-share COM1 Share#1 out “C:\Program Files (x86)\Eltima Software\Advanced Serial Port Terminal” rwc*

UNSET-SHARE "port" "bundle" [in|out] "app name"
Removes an application from the list of applications that are able to access the shared port.

> ***Example:***
> *vspdpro_service.exe unset-share COM1 Share#1 out "C:\Program Files (x86)\Eltima Software\Advanced Serial Port Terminal"*

SET-WIRES "bundle" dtr-pins rts-pins
Sets a custom signal lines pinout.

The parameters **dtr-pins, rts-pins** should contain whole decimal numbers:

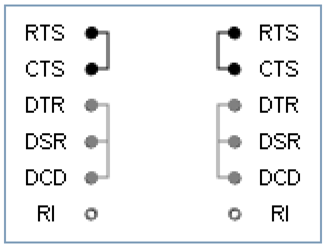
0 - none
1 - CTS
2 - DSR
3 - CTS, DSR
4 - RI
5 - CTS, RI
6 - DSR, RI
7 - CTS, DSR, RI
8 - DCD
9 - CTS, DCD
10 - DSR, DCD
11 - CTS, DSR, DCD
12 - RI, DCD

13 - CTS, RI, DCD
14 - DSR, RI, DCD
15 - CTS, DSR, RI, DCD

> ***Example:***
> *vspdpro_service.exe set-wires Pair#1 1 11*
> *//Set a custom pinout for the bundle Pair#1:* DTR->CTS; RTS->CTS, DSR, DCD.

To set up a loopback pinout scheme, you should use the parameters 0 0.

> ***Example:***
> *vspdpro_service.exe set-wires Pair#1 0 0*
> *//Set a loopback wiring scheme for the port bundle Pair#1: DTR->DSR and RI, RTS->CTS*



4. Additional options

RESTORE file [backup-file]
Restores bundle settings.

You can save current bundle settings to a file and then restore the settings when required.

> ***Example:***
> *vspdpro_service.exe restore C:\vspd_config.json C:\backup-file.json*

BACKUP file
Saves bundle settings to a file.

> ***Example:***
> *vspdpro_service.exe backup C:\vspd_config.json*

<u>**Examples of commands to create serial port bundles.**</u>

Serial Port Pair:

```
vspdpro_service.exe add-bundle Pair#1 pair
vspdpro_service.exe add-port COM1 Pair#1 in virtual
vspdpro_service.exe add-port COM2 Pair#1 out virtual
```

\\Create a virtual port pair with the port COM1 on the Input side and the port COM2 on the Output side.

Split Port Bundle:

```
vspdpro_service.exe add-bundle Split#1 split
vspdpro_service.exe add-port COM1 Split#1 in real
vspdpro_service.exe add-port COM20 Split#1 out virtual
vspdpro_service.exe add-port COM21 Split#1 out virtual
```

\\Create a bundle Split#1 with one real serial port on the Input side and two (or more) virtual ports on the Output side.

Joined Port Bundle:

```
vspdpro_service.exe add-bundle Join#1 join
vspdpro_service.exe add-port COM1 Join#1 in real
vspdpro_service.exe add-port COM2 Join#1 in real
vspdpro_service.exe add-port COM19 Join#1 in virtual
```

\\Create a bundle Join#1 with two (or more) real serial ports on the Input side and one virtual port on the Output side.

Redirection Port Bundle:

```
vspdpro_service.exe add-bundle Redirect#1 redirect
vspdpro_service.exe add-port COM1 Redirect#1 in real
vspdpro_service.exe add-port COM15 Redirect#1 out virtual
or
vspdpro_service.exe add-port COM2 Redirect#1 out real
```

\\Create a bundle Redirect#1 with one real port on the Input side and one real or virtual port on the Output side.

Shared Port Bundle:

```
vspdpro_service.exe add-bundle Share#1 share
vspdpro_service.exe add-port COM1 Share#1 in real
vspdpro_service.exe add-port COM1 Share#1 out share
```

\\*Create a bundle Share#1 with one real port COM1 on the Input side and one shared port with the same name on the Output side.*

Complex Port Bundle:

```
vspdpro_service.exe add-bundle Complex#1 complex
vspdpro_service.exe add-port COM1 Complex#1 in real
vspdpro_service.exe add-port COM10 Complex#1 in share
vspdpro_service.exe add-port COM11 Complex#1 in virtual
vspdpro_service.exe add-port COM2 Complex#1 out real
vspdpro_service.exe add-port COM12 Complex#1 out share
vspdpro_service.exe add-port COM13 Complex#1 out virtual
```

\\*Create a bundle Complex#1 with multiple real, virtual, and shared ports on the Input and Output sides.*

Port Switcher Bundle:

```
vspdpro_service.exe add-bundle Switch#1 switcher
vspdpro_service.exe add-port COM20 Switch#1 in switch
vspdpro_service.exe add-port COM1 Switch#1 out real
vspdpro_service.exe add-port COM2 Switch#1 out real
```

\\*Create a bundle Switch#1 with a switcher port named COM20 on the Input side and multiple real ports on the Output side.*

Merged Port Bundle:

```
vspdpro_service.exe add-bundle Merge#1 merge
vspdpro_service.exe add-port COM2 Merge#1 in real
vspdpro_service.exe add-port COM10 Merge#1 in virtual
vspdpro_service.exe add-port COM11 Merge#1 in virtual
```

\\*Create a bundle Merge#1 with multiple real and virtual ports on the Input side.*

Loopback Port Bundle:

```
vspdpro_service.exe add-bundle Loopback#1 loopback
vspdpro_service.exe add-port COM10 Loopback#1 in vspd
```

\\*Create a bundle Loopback#1 with one “vspd” port on the Input side.*

Adding Virtual Serial Port Driver functionality to your application

vspdpro.dll is a dynamic link library that allows you to control serial ports either on a local or remote machine directly from your own application.

This lets you create and delete bundles, add to or remove ports from bundles, set the main port, grant access rights to applications, etc. on the go without using standard configuration utility with GUI.

vspdpro.dll should be placed in the same folder as the application which uses it.

If you develop a 64-bit application, you should use the ***vspdpro64.dll*** file instead. This file is supplied with Virtual Serial Port Driver. *vspdpro64.dll* should be renamed to *vspdpro.dll*.

Please note: When the Serial Port Driver Service is managed on the remote machine,
the SharedGetPortApplications function won't work.

Functions

SSGetLastError

DWORD __stdcall SSGetLastError();

Routine Description:

Returns the code of the last error.

Return Value:

The code of the last error.

ConnectToLocalService

BOOL __stdcall ConnectToLocalService();

Routine Description:

Establishes connection with VSPD Service on the local machine for bundle creation and ports management.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

ConnectToRemoteService

BOOL __stdcall ConnectToRemoteService(CComVariant &host, CComVariant &login, CComVariant &password);

Routine Description:

Establishes connection with the Virtual Serial Port Driver Service on the remote machine for bundles creation and ports management.

Arguments:

host - remote host name or its IP

login - administrator's account login to connect to the remote machine. If not set, configuration won't be kept in Windows Registry.

password - administrator's account password

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

DisconnectFromService

BOOL __stdcall DisconnectFromService();

Routine Description:

Disconnects from the VSPD Service, either on the remote or local machine.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

AddBundle

```
BOOL __stdcall AddBundle(CComVariant &BundleName);
```

Routine Description:

Creates a new ports bundle.

Arguments:

BundleName - the name of a new bundle

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

AddBundleEx

BOOL __stdcall AddBundle(CComVariant &BundleName, BundleType Type);

Routine Description:

Creates a new port bundle.

Arguments:

BundleName - the name of bundle to create

BundleType - defines bundle behavior

The following are the recommended number and types of ports to use for correct bundle operation based on the bundle type:

Bundle type	**Value**	**Input port(s)**	**Output port(s)**
bundleSplit	0	1 real port	1 or more virtual ports
bundleJoin	1	1 or more real ports	1 virtual port
bundleRedirect	2	1 real port	1 real or virtual port
bundleShare	3	1 real port	1 shared port
bundleAdvanced	4	1 or more real, virtual or shared ports	1 or more real, virtual or shared ports
bundleSwitch	5	1 switcher port	1 or more real ports
bundleMerge	6	2 or more real or virtual ports	No ports
bundlePair	7	1 virtual port	1 virtual port
bundleLoopback	8	1 virtual port	No ports

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetType

BOOL __stdcall BundleGetType(CComVariant &BundleName, BundleType & Type);

Routine Description:

Gets bundle type.

Arguments:

BundleName - the name of the bundle to be checked.

BundleType - defines bundle behavior.

Bundle Type	Value	Description
bundleSplit	0	Used to split one real port into several virtual ones
bundleJoin	1	Used to join several real ports into one virtual
bundleRedirect	2	Used to redirect data from a real port to another real or virtual port
bundleShare	3	Used to share a real port among several applications
bundleAdvanced	4	Used to split, join and/or share serial data streams
bundleSwitch	5	Used to automatically switch between several real ports
bundleMerge	6	Used to merge several real and/or virtual ports
bundlePair	7	Used to pair virtual ports
bundleLoopback	8	Used to create a loopback connection via a virtual serial port

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

DeleteBundle

BOOL __stdcall DeleteBundle(CComVariant &BundleName);

Routine Description:

Deletes a port bundle from the system.

Arguments:

BundleName - the name of the bundle to be deleted.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

GetBundleState

BOOL __stdcall GetBundleState(CComVariant &BundleName);

Routine Description:

Gets the current state of a port bundle.

Arguments:

BundleName - the name of the bundle to be checked.

Return Value:

True - if the bundle exists. Otherwise, the "False" value is returned.

BundleAddPort

BOOL __stdcall BundleGetPortType(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant *PortType);

Routine Description:

Adds a port to a bundle.

Arguments:

BundleName - the name of the bundle to add a port to

Side - defines IN or OUT side of the bundle:

0 = IN

1 = OUT

PortName - the name of the port to be added to the bundle

Type - the type of the port to be added to the bundle:

1 = Real

2 = Virtual

4 = Shared

8 = Switcher

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleDeletePort

BOOL __stdcall BundleDeletePort(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Removes a port from a bundle.

Arguments:

BundleName - the name of the bundle to remove a port from

Side - defines IN or OUT side of the bundle:

0 = IN

1 = OUT

PortName - the name of the port to be removed from the bundle

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetPortState

BOOL __stdcall BundleGetPortState(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Gets the port's current status and state in the specified bundle.

Arguments:

BundleName - the name of the bundle

Side - defines IN or OUT side of the bundle:

0 = IN

1 = OUT

PortName - the name of the port to be checked.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

GetBundlesNames

BOOL __stdcall GetBundlesNames(CComVariant *BundlesNames);

Routine Description:

Gets the list of all bundles existing in the system.

Arguments:

*BundlesNames - a pointer to a variable (of "Variant" type) to which the array of strings containing the list of bundles is returned.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

GetPortsNamesByBundleAndSide

BOOL __stdcall GetPortsNamesByBundleAndSide(CComVariant &BundleName, PortSide Side, CComVariant *PortNames);

Routine Description:

Gets the list of ports in the specified bundle (with the bundle side indicated).

Arguments:

BundleName - the name of the bundle

Side - defines IN or OUT side of the bundle:

0 = IN

1 = OUT

*PortNames - a pointer to a variable (of "Variant" type) to which the array of strings containing the list of ports' names is returned.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleSetMainPortForSide

BOOL __stdcall BundleSetMainPortForSide(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Sets the main port on the specified side of the bundle.

Arguments:

BundleName - the name of the bundle

Side - the bundle side where a port has to be set as the main one:

0 = IN

1 = OUT

PortName - the name of the port to be set as the main one

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetMainPortForSide

BOOL __stdcall BundleGetMainPortForSide(CComVariant &BundleName, PortSide Side, CComVariant *PortName);

Routine Description:

Gets the name of the main port on the specified side of the bundle.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = IN

1 = OUT

*PortName - a pointer to a variable (of "Variant" type) to which a variable of "String" type containing the name of the main port is returned.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetBaudrateEmulation

BOOL __stdcall BundleGetBaudRateEmulation(CComVariant &BundleName, BOOL & Value);

Routine Description:

Indicates whether baudrate emulation for virtual ports in the specified bundle is enabled or disabled.

Arguments:

BundleName - the name of the bundle to be checked

*Value - a pointer to a BOOL variable that will be set to true if baudrate emulation is enabled; false otherwise.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleSetBaudrateEmulation

BOOL __stdcall BundleSetBaudRateEmulation(CComVariant& BundleName, BOOL value);

Routine Description:

Enables/disables baudrate emulation for virtual ports in the specified bundle.

Arguments:

BundleName - the name of the bundle where baudrate emulation will be enabled or

disabled. Value - the baudrate emulation option state to be set (enabled/disabled)

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetWiring

BOOL __stdcall BundleGetWiring(CComVariant& BundleName, SignalLine& dtr, SignalLine& rts);

Routine Description:

Gets internal wiring (pinout) of a virtual serial port pair. Applicable only to Pair and Loopback bundles.

Arguments:

BundleName - the name of the bundle to be checked

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleSetWiring

BOOL __stdcall BundleSetWiring(CComVariant& BundleName, SignalLine dtr, SignalLine rts);

Sets internal wiring (pinout) for a virtual serial port pair. Applicable only to Pair and Loopback bundles.

Arguments:

BundleName - the name of the bundle

Dtr - specifies signal lines to be connected to the DTR line

Rts - specifies signal lines to be connected to the RTS line

SignalLine type is a flag enum with the following values:

CTS = 1,

DSR = 2,

RING = 4,

DCD = 8

True - if the function succeeds. Otherwise the "False" value is returned.

GetRealPortsList

BOOL __stdcall GetRealPortsList(CComVariant *PortNames);

Routine Description:

Returns the list of real ports present in the system.

Arguments:

*PortNames - a pointer to a variable (of "Variant" type) to which a variable of "Array" type containing the list of real ports is returned.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetPortType

BOOL __stdcall BundleAddPort(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant Type);

Routine Description:

Gets port type.

Arguments:

BundleName - the name of the bundle containing the port to be checked

Side - the bundle side:

0 = In

1 = Out

PortName - the name of the port which type is to be checked.

*PortType - a pointer to a variable (of "Variant" type) to which the type of the port is returned.

Return Value:

Type of the port:

Real = 1

Virtual = 2

Shared = 4

Switcher= 8

BundleSetRealPortParameters

BOOL __stdcall BundleSetRealPortParameters(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant baudrate, CComVariant parity, CComVariant databits, CComVariant stopbits, CComVariant &flow);

Routine Description:

Sets real port parameters.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName- the real port name

baudrate - baudarte value (digit)

parity - parity value:

NO PARITY = 0

ODD PARITY = 1

EVEN PARITY = 2

MARK PARITY = 3

SPACE PARITY = 4

databits - databits value (digit)

stopbits - stopbits value (digit)

flow - flow control value:

None = 0
Hardware = 2

Xon/Xoff = 1

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleSetDynamicParametersForRealPort

BOOL __stdcall BundleSetDynamicParametersForRealPort(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Defines that virtual or shared port set as main in the bundle will manage parameters of real port on the other side of the bundle (that is, real port will have dynamic parameters).

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName - the real port name

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleGetRealPortParameters

BOOL __stdcall BundleGetRealPortParameters(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant *baudrate, CComVariant *parity, CComVariant *databits, CComVariant *stopbits, CComVariant *flow);

Routine Description:

Gets real port parameters.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName- the real port name

baudrate - baudarte value (digit)

parity - parity value:

NO PARITY = 0

ODD PARITY = 1

EVEN PARITY = 2

MARK PARITY = 3

SPACE PARITY = 4

databits - databits value (digit)

stopbits - stopbits value (digit)

flow - flow control value:

None = 0
Hardware = 2

Xon/Xoff = 1

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundlePortGetPermissions

BOOL BundlePortGetPermissions(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant *Permission);

Routine Description:

Gets permissions for real or virtual port.

Arguments:

BundleName - the name of the bundle.

Side - the bundle side:

0 = In

1 = Out

PortName - the real/virtual port name.

Permission:

1 = Read from other direction.

2 = Write to other direction.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundlePortSetPermissions

BOOL BundlePortSetPermissions(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant Permission);

Routine Description:

Sets permissions for a real or virtual port.

Arguments:

BundleName - the name of the bundle.

Side - the bundle side:

0 = In

1 = Out

PortName - the real/virtual port name.

Permission:

1 = Read from other direction.

2 = Write to other direction.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

BundleRealPortAreParametersDynamic

BOOL __stdcall BundleRealPortAreParametersDynamic(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Verifies whether real port has dynamic parameters (that is, a virtual or shared port, set as main in bundle, manages parameters of the real port).

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName - the real port name

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

SharedDelAllPermissions

BOOL __stdcall SharedDelAllPermissions(CComVariant &BundleName, PortSide Side, CComVariant &PortName);

Routine Description:

Defines that neither of applications can manage shared port parameters.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName - the shared port name

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

SharedSetDefaultPermissions

BOOL __stdcall SharedSetDefaultPermissions(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant &AppPath);

Routine Description:

Sets default access rights (read/write) for the application which will manage shared port parameters.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName- the shared port name

AppPath- the path to the application which will manage shared port parameters

Return Value:

True - if the function succeeds. Otherwise the "False" value is returned.

SharedGetPortApplications

BOOL __stdcall SharedGetPortApplications(CComVariant &PortName, CComVariant *Applications);

Routine Description:

Gets the list of applications that opened shared port.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

    0 = In

    1 = Out

PortName - the shared port name

Applications - a pointer to a variable to which the array of strings is returned. "Application" variable contains an array of "Variant" type.

Each array element contains array of "variant" type (of 4-string length) with the following parameters:

- 0 - application's name, string
- 1 - application's permissions, digit:
  - Read = 1
  - Write = 2
  - Control = 4
- 2 - number of input data in bytes
- 3 - number of written data in bytes

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

SharedPortSetPermissionsForApp

BOOL __stdcall SharedPortSetPermissionsForApp(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant &AppPath, CComVariant Permission);

Routine Description:

Sets application's access rights to the shared port.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName- the shared port name

AppPath - the path to the application which will open the shared port

Permission - can have one of the following meanings:

Read = 1

Write = 2

Control = 4

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

SharedPortGetAppPermissions

BOOL __stdcall SharedPortGetAppPermissions(CComVariant &BundleName, PortSide Side, CComVariant &PortName, CComVariant *AppPaths, CComVariant *Permission);

Routine Description:

Gets list of applications with custom access rights to the specified shared port.

Arguments:

BundleName - the name of the bundle

Side - the bundle side:

0 = In

1 = Out

PortName- the shared port name

*AppPaths - a pointer to a variable (of "Variant" type) containing paths to the applications with custom access rights

Permission - can have one of the following meanings:

Read = 1

Write = 2

Control = 4

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

VirtualGetAccessList

BOOL __stdcall VirtualGetAccessList(CComVariant& BundleName, PortSide side, CComVariant& PortName, CComVariant* acl);

Routine Description:

Gets the list of application masks set for the specified shared port.

Arguments:

BundleName - the name of the bundle containing the shared port

Side - the bundle side:

0 = In

1 = Out

PortName- the shared port name

*acl - a pointer to a VARIANT variable that contains the paths to application access masks.

An access mask is a string of characters that indicates whether or not a specified file (application) can access the shared port. The string starts either with ‘+’ (access granted) or ‘-’ (access denied) followed by the file name. The access mask string may also contain standard glob patterns with wildcard characters like ‘*’ or ‘?’.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

VirtualSetAccessList

BOOL __stdcall VirtualSetAccessList(CComVariant& BundleName, PortSide side, CComVariant& PortName, CComVariant& acl);

Routine Description:

Sets the list of application masks for the specified virtual port.

Arguments:

BundleName - the name of the bundle containing the shared port

Side - the bundle side:

0 = In

1 = Out

PortName- the shared port name

*acl - a pointer to a VARIANT variable that contains the paths to application access masks.

An access mask is a string of characters that indicates whether or not a specified file (application) can access the shared port. The string starts either with '+' (access granted) or '-' (access denied) followed by the file name. The access mask string may also contain standard glob patterns with wildcard characters like '*' or '?'.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

GetCurrentUserSession

```
BOOL __stdcall GetCurrentUserSession(CComVariant * UserSession);
```

Routine Description:

Gets current user session identifier.

PairGetUserSession

```
BOOL __stdcall PairGetUserSession(CComVariant& BundleName, CComVariant* UserSession);
```

Routine Description:

Gets the identifier of the user session the specified port pair is created for.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

PairSetUserSession

```
BOOL __stdcall PairSetUserSession(CComVariant& BundleName, CComVariant& UserSession);
```

Routine Description:

Links a virtual port pair to a specific user session. The pair will be created for the given user session and will not be visible for other user sessions.

Note: This function needs to be called before adding any ports to your Pair bundle.

If you need to change the user session for an existing port pair, you have to delete ports from the pair and add them again to apply the changes.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

OptionSetOnlyMainPortCanSend

BOOL __stdcall OptionSetOnlyMainPortCanSend(BOOL value);

Routine Description:

Sets that only the main port is allowed to send data to the other side of the bundle.

Return Value:

True - if the function succeeds. Otherwise the "False" value is returned.

OptionSetIgnoreErrors

```
BOOL __stdcall OptionSetIgnoreErrors(BOOL value);
```

Routine Description:

Sets to ignore errors at port creation or opening.

Return Value:

True - if the function succeeds. Otherwise the "False" value is returned.

OptionSetAutoOpenPort

BOOL __stdcall OptionSetAutoOpenPort(BOOL value);

Routine Description:

Sets to open ports automatically after they appear in the system. Can be useful for COM ports that appear in the system via USB adapters.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

OptionSetBaudrateEmulation

```
BOOL __stdcall OptionSetBaudrateEmulation(BOOL value);
```

Routine Description:

Enables baudrate emulation for virtual ports.

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

OptionSetPortToPrint

BOOL __stdcall OptionSetPortToPrint(BOOL value);

Routine Description:

Sets whether virtual or shared ports can be available for a printing pool (a group of printers attached to a print queue).

Return Value:

True - if the function succeeds. Otherwise, the "False" value is returned.

OptionGetIfSendToMainSet

```
BOOL __stdcall OptionGetIfSendToMainSet(BOOL &value);
```

Routine Description:

Verifies whether only the main port can transfer data to the other side of the bundle.

Return Value:

"True" if only the main port can transfer data to the other side of the bundle. Otherwise, the "False" value is returned.

OptionGetIgnoreErrorsOnPortCreation

BOOL__stdcall OptionGetIgnoreErrorsOnPortCreation(BOOL &value);

Routine Description:

Gets whether errors on port creation are ignored.

Return Value:

"True" if errors on port creation are ignored. Otherwise, the "False" value is returned.

OptionGetPortsAutoOpen

```
BOOL __stdcall OptionGetPortsAutoOpen(BOOL &value);
```

Routine Description:

Gets whether ports are opened automatically after they appear in the system.

Return Value:

"True" if ports are opened automatically after they appear in the system. Otherwise, the "False" value is returned.

OptionGetBaudrateEmulation

BOOL __stdcall OptionGetBaudrateEmulation(BOOL &value);

Routine Description:

Gets whether baudrate emulation for virtual ports is enabled.

Return Value:

"True" if baudrate emulation for virtual ports is enabled. Otherwise, "False" value is returned.

OptionGetPortToPrint

BOOL __stdcall OptionGetPortToPrint(BOOL &value);

Routine Description:

Gets whether virtual and shared ports can be added to a printing pool.

Return Value:

"True" if virtual and shared ports are allowed to be added to a printing pool. Otherwise, the "False" value is returned.

BundleUpdateSettings

BOOL __stdcall BundleUpdateSettings();

Routine Description:

Lets the VSPD Service apply new settings to the currently available bundles.

Return Value:

True - if the function succeeds. Otherwise "False" value is returned.

Troubleshooting

- **How can I set the main port in a bundle?**

The main port is a port that is allowed to manage control lines (RTS/CTS, DSR/DTR, DCD, RING) on the other side of the bundle.

1. When splitting a real serial port, the real port is always main on the “IN” side of the bundle.
2. When joining several real ports, the virtual port is always main on the “OUT” side of the bundle.
3. When redirecting serial traffic from any real port to another port, ports of both sides are the main ones.
4. When sharing a real port between several applications, the ports of both sides are the main ones. The application with "Control" access rights will manage port parameters provided it was the first to open the port. If no application in a bundle has "Control" access rights, then neither of them will be able to manage shared port parameters.
5. In a complex bundle main port is always set manually. If it is not done, then there will be no main port in the complex bundle. A virtual port, set as main in a complex bundle, manages the dynamic parameters of a real port on the other side of the bundle. If a shared port is set as main in a complex bundle, any application with "Control" access rights will manage real ports parameters.

Note: You can find more information on dynamic port parameters in the "How can I set custom port settings?" section below on this page.

- **How can I set custom real port settings?**

PortSettings parameters could be empty, which means that dynamic connection parameters will be used instead. In that case, real port parameters will depend on the parameters of the main port on the opposite side. To transfer all parameters the opposite main port should be of Virtual or Share type. If a port is of Real type, it will open with the last applied parameters.

However, you can set port settings manually. Choose the "Use custom settings" option (Toolbar-->Port settings) which allows you to set different values such as:

- **Baudrate**

Possible baudrate values are: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000 and 256000 bauds. 9600 is selected by default.
You may also put any custom baudrate value that you need, for example, 150000. Place a mouse cursor anywhere inside this field and hit the `Backspace" button.

Alternatively, you may set custom baudrate value via the registry (for SDK license owners). For example:

```
"COM1 [Communications Port]"="Real|15000,O,6,2,X"
```

Custom port settings are set as follows: baudrate - 15000, parity - odd, databits - 6, 2 stopbits, flow control - Xon/Xoff.

To get more information about working with Windows registry you should get the SDK license.

- **Parity**

The parity bit in each character can be set to "No parity", "Odd", "Even", "Mark" and "Space". Default value is "No parity".

- **Databits**

Possible values are: 5, 6, 7 and 8. "7" is selected by default.

- **Stopbits**

Possible values are: "1 stop bit" and "2 stop bits". Default value is "1 stop bit".

- **Flow control**

Possible values are: "None", "Hardware", "Xon/Xoff". Default value is "None".

Renaming virtual serial ports

By default, Virtual Serial Port Driver names created virtual serial ports as follows: Electronic Team Virtual Serial Port (COM 3). To know the exact port name, look in Device Manager.

We are glad to offer you the possibility to change the names of virtual serial ports, created by Virtual Serial Port Driver. Below we give you the command for renaming Electronic Team Virtual Serial Ports into Electronic Team Software Virtual Serial Ports to let you see the procedure and learn how to do it easily.

So, you can test the ports renaming feature using the following command:

```
Vspdxp_install.exe /c "FE50Eltima Software Virtual Serial Port"
```

By default, *vspdxp_install.exe* is located in the folder where you installed Virtual Serial Port Driver (C:\Program Files\Electronic Team\Virtual Serial Port Driver 10\).

NOTE: The code (4 symbols) uniquely corresponds to the custom name (in our example to "Electronic Team Software Virtual Serial Port").

If you execute this command with any other four digits or name, the port's name will be set to default (Electronic Team Virtual Serial Port). If you would like to have the ports renamed, start with sending us the name you want your ports to have. We will generate the code which is bound to this name and send it back to you. We assume that the SDK users will be especially interested in this option. The renaming ports option costs EUR 200/USD 300.

When placing the order, enter the full name for the virtual port (the one you want to show up in the Device Manager) when asked for additional details or comments.