

IPm File Loader Utility (sxfileload.exe)

Abstract:

This Windows utility downloads or uploads files into or out of an IPm controller. This loader may be called from any application that can execute (run) a program. The transfer operation is defined by command line arguments included in the command. This flexible utility accepts wild cards (to copy groups of files), reads files from the station, and can install tar files (so that scripts can be run in the IPm). This execution of this utility is transparent (no Graphic User Interface is presented) with return information presented in a status file. Files may be transferred over Ethernet or serial links (basically any communication link supported by the I/O Tool Kit).

Software requirements:

The sxfileload.exe is supplied with the I/O Tool Kit, which must be installed for this utility to function. (This utility uses various Tool Kit resources to run.) The Tool Kit itself need not be running. This utility requires that a valid Remote Access" option be licensed as part of the I/O Tool Kit registration in the Windows computer that is executing this command.

This utility will transfer files into or out of any Sixnet controller or RTU that contains an IPm engine. This includes the VT-IPM, ST-IPM, ST-GT-1210, ET-GT-ST-3, all OEM derivatives of these products, and of course all future products that are IPm based.

Using this utility:

This utility may be called from Windows applications and scripts running within a Windows environment. It is called in the manner that any run command would be executed. Command line arguments specified as part of the command define the file transfer to occur and the communications path to use. This utility will run blind (not display any GUI or visible indication of its operation).

To transfer multiple files with a single transfer command, wildcards may be specified. When using multiple instances of the sxfileload.exe program, use the -Instance switch to assign an identifier to each call to sxfileload.exe. The calling application should use unique identifiers (numbers or strings). The identifier is then used by the calling application to read the exit code from the sxstatus.ini file. At this time, the calling application should remove the entry from the sxstatus.ini file to prevent the size of the file from becoming excessively large. If the sxfileload.exe program is not used as a multiple instance program, the calling application does not need to remove entries from the status file.

Return codes are available to the calling application by either reading the exit code directly or by reading from the sxstatus.ini file. This file is located in the Windows root directory. A list of valid return codes may be found in Appendix 1 of this Technical Note. A sample of an sxstatus.ini file is shown below:

```
[STATUS]
sixlog=-1
sixdial=0
sxfileload=0
```

Other applications write to this file using their own entry names (sixlog, sixdial, ...). The entry used for the sxfileload.exe program is "sxfileload". If multiple instances are used and the -Instance switch is used, the identifier will be used as the entry in the [STATUS] section. The return codes of sxfileload.exe are listed the end of this document.

This utility transfers files using Sixnet Universal protocol. Since ftp (file transfer protocol) is not used, it may be disabled in the IPm station (or not installed at all, which is the case for many products that have an embedded IPm engine but do not have an "IPM" in their part number). This feature is a major security benefit since ftp could be potentially used by unauthorized parties to access the station's file system.

Command line arguments:

Here are the SxFileLoad command line arguments that define the file transfer. The applicable arguments will depend on the type of communication path being used. The SxFileLoad.exe filename and the source file's filename must each include the appropriate path and quotes, as shown in the examples at the bottom of this page. Note that these command line arguments are not case sensitive. The capital letters show the required letters - lower case letters are optional. These options are not case sensitive, except for the specification of path and file information (Linux file names are case sensitive), and COM port designations in the DEvice argument..

DEvice=(COM1, COM3, Ethernet, ...) Required argument

Specify COMx for serial port communication, where "COM" must be capitalized, and x is a port # (1 through 255). Specify Ethernet for Ethernet communication. Also specify the -IPAddr option (see below).

IPAddr=(destination IP address) Required for Ethernet communications

Specify the IP address of the passthru station or direct station when using Ethernet communication; ie/ 10.1.0.1.

BAud=(9600, 19200, ...) Required for serial communications

Specify which baud rate to use for direct serial communication. The choices are: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600

MDe=(8N1N, 8E1H, ...) Optional - "8N1N" defaults will apply

Where the first character is the number of data bits, the second character is parity, the third character is the number of stop bits, and the fourth is flow control (N = none, H = hardware, X = xon/xoff). Specify which data bits, stop bits, parity and handshaking to use for direct serial communication. If not specified, 8 data bits, no parity, one stop bit and no handshaking will be used.

The choices are:

7 or 8 data bits

No , Even, Mark, Odd or Space parity

1 or 2 stop bits

None, Hardware or Xon/Xoff handshaking

STAnum (destination station number) Required argument

Specify the address of the destination station whether it is being accessed directly or through a passthru station.

Note: Passthru allows you send a message to one station with the expectation that it will be forwarded to another station. The most common application is passing communications over an Ethernet (or any IP link) and then out a serial link to another station. Specify the IP address of the relaying station and the station number of the final destination station.

Operation (operation to perform) Default = "copy files"

"copy_files" = copy files, "install_tar" = install a tar file, "read_file" = read a file from the station). The default option if this argument is not specified is to copy files. When reading files, the source file is the file in the station and the destination file is the file on the PC. Both file names must be fully specified when reading files. No wildcards are allowed when reading files from the station.

SOURCEfile (source file name – wildcards allowed when copying/installing files) Required argument
This required switch is where you specify the path and name of the file to load into the destination IPm-based station.

The path must be included along with the name. Path names and file names may need to be placed in quotes. When reading files from the station, this is the name of the file in the station. Wildcards are not allowed when reading files from the station.

DESTfile (destination file name or path) Required argument

This required switch is where you specify the destination folder and/or file in the destination IPm-based station. The path may need to be placed in quotes. When reading files from the station, this is the name of the file on the PC.

When installing tar files, this is the path that the installation should be done relative to. Usually, this would be the root directory "/". If the tar file has an /install/doinst.sh file within it, the station will execute this script after unpacking the contents of the tar file. This provides a way for users to install "packages" that perform further installation operations beyond the unpacking of the tar file.

Note: Although this utility itself is not case sensitive, it will preserve the case of the path and file names specified in the SOURCEfile and DESTfile arguments. Remember that the IPM is Linux based and therefore file names in the IPm are case sensitive.

INSTANCE (instance identifier assigned to the executed sxfileload.exe) Optional argument

Use this switch when running multiple instances of the sxfileload.exe program. The calling application can assign a number or string to this field. The identifier is then used as the entry in the sxstatus.ini file to record the exit code of the sxfileload.exe execution. It is the responsibility of the calling application to keep the identifiers unique. The calling application may also need to remove the entry after reading the return code to prevent the status file from becoming excessively large.

OWNER (Linux owner name of destination file(s)) Optional argument

This optional switch may be used when installing a tar file or copying files. When used with the install option, the unpacking of the tar file is done as if the owner was logged in.

GROUP (Linux group name of destination files(s)) Optional argument

This optional switch may be used when copying files. The destination file is assigned the group ID after it is copied into the station.

PERmissions (Linux permissions value of destination file) Optional argument

This optional switch may be used when copying files. The destination file(s) will be given the permissions as specified. Permissions must be specified in octal format (755, 4755, 644, ...).

PREServe (preserves tar permissions when used with "install_tar") Optional argument

Y or y = preserve the settings as stored in the tar file. N or n = do not preserve the permissions stored in the tar file. If this option is not specified, permissions will not be preserved. This option is only for use when installing tar files.

Sample command lines:

Example command line for file transfer directly to a remote station using COM1 of the computer:

```
"C:\Sixnet Tools\Programs\sxfileload.exe" -SOURCE="C:\mypath\my_recipe.txt" -DEV=COM1 -  
MODE=8N1H -BAUD=4800 -DEST="/usr/local/bin/my_recipe.txt" -STA=4
```

Example command line for file transfer using an Ethernet-to-serial passthru connection:

```
"C:\Sixnet Tools\Programs\sxfileload.exe" -SOURCE="C:\mypath\my_recipe.txt" -DEV=Ethernet -IP=10.1.0.1 -  
STA=4 -DEST="/usr/local/bin/my_recipe.txt"
```

Example command line for installing a tar file relative to the root directory:

```
"C:\Sixnet Tools\Programs\sxfileload.exe" -SOURCE="C:\data files\ipmcfg.tar.gz" -DEST="/" -DEV=Ethernet -  
IP=10.1.0.18 -STA=18 -PRESERVE=y -OPERATION="install_tar"
```

Example command line for copying multiple files into the station using wildcards/pattern matching:

```
"C:\Sixnet Tools\Programs\sxfileload.exe" -SOURCE="C:\data files\ulb.*" -DEST="/usr/local/bin/*" -  
DEV=Ethernet -IP=10.1.0.18 -STA=18
```

In the above example, if the following files exist on the PC:

```
c:\data files\ulb.file1.txt  
c:\data files\ulb.file2.txt  
c:\data files\ulb.file3.txt
```

The result would be the creation/copying of:

```
/usr/local/bin/file1.txt  
/usr/local/bin/file2.txt  
/usr/local/bin/file3.txt
```

Note that the Linux destination filenames end up with all lower case letters. If the Windows source filenames have one or more upper case characters and the Linux destination filenames must match exactly, you'll need to list the filenames individually instead of specifying wildcard names.

Further explanation of pattern matching may be found in Appendix 2 at the end of this Technical Note.

Appendix 1: Error codes

Error codes (if any) are written to the file sxstatus.ini in your Windows root directory. Here are the possible error codes:

1 Operation in progress (program started the operation but has not completed yet)

0 Successful completion (no errors)

-1 Invalid port specified

-2 Invalid baud specified

-3 Invalid mode specified (for COM ports only)

-4 Invalid IP specified (Ethernet only)

-5 Invalid station number

-6 Source file not specified

-7 Destination file not specified

-8 Unable to read source file

-9 Unable to open communications

-10 Failed to communicate

-11 Unable to rename a file

-12 Invalid option specified (user specified an option that doesn't work with other options specified)

-13 Invalid permissions specified

-14 Invalid operation type specified

-15 Multiple wildcards specified in destination

-16 Wildcard mismatch

-17 Invalid filename

-18 File not found

-19 Unable to access destination file

-20 Unable to access source file

-21 Out of memory on PC (when reading files from the station)

-22 Unsuccessful with regards to changing owner permissions

-23 Unsuccessful with regards to changing group permissions

-24 Unsuccessful with regards to changing the access mode of one or more files

-99 Program not registered

Appendix 2: Pattern Matching in Linux

The Linux operating system in IPm-based controllers does not interpret the * symbol the same way Windows does. Here are some examples of how this works:

Source Destination

demoapp\ulb.* /usr/local/bin/*

if these files exist in the Windows folder c:\sixnet tools\projects\demoapp\:

ulb.file1.txt

ulb.file2.txt

ULB.FILE3.txt

then they would be loaded into an IPm station as:

/usr/local/bin/file1.txt

/usr/local/bin/file2.txt

/usr/local/bin/file3.txt

Note that the Linux destination filenames end up with all lower case letters. If the Windows source filenames have one or more upper case characters and the Linux destination filenames must match exactly, you'll need to list the filenames individually instead of specifying wildcard names.

If you wanted the filename to stay the same, then you would specify this:

Source Destination

demoapp\ulb.* /usr/local/bin/**ulb.***

resulting in:

/usr/local/bin/ulb.file1.txt

/usr/local/bin/ulb.file2.txt

/usr/local/bin/ulb.file3.txt

The * character is used to match patterns in Linux. The * character is used as a wildcard in Windows. When entering the source file location, use the Windows scheme. When entering the IPm destination, be aware that the pattern matching scheme should be used instead of the Windows wildcard scheme.

Pattern matching allows you to place a prefix in front of the files you want to go to a certain directory in the station. In the example above, "ulb." refers to "usr/local/bin/". This naming convention is not something that we enforce, and users can come up with anything they want to help organize their files as long as it works with the pattern matching.