

Using External Configuration Utilities with the I/O Tool Kit

Abstract:

OEMs create Windows configuration utilities for the I/O drivers and applications programs they supply to run in an IPm (Linux-based) controller. By including these user-defined files on the station's "Files to Load" list and/or the "User Software" list, the process of loading a complete project, including the OEM features is automated. This document explains the interface between an OEM's configuration utility and the I/O Tool Kit.

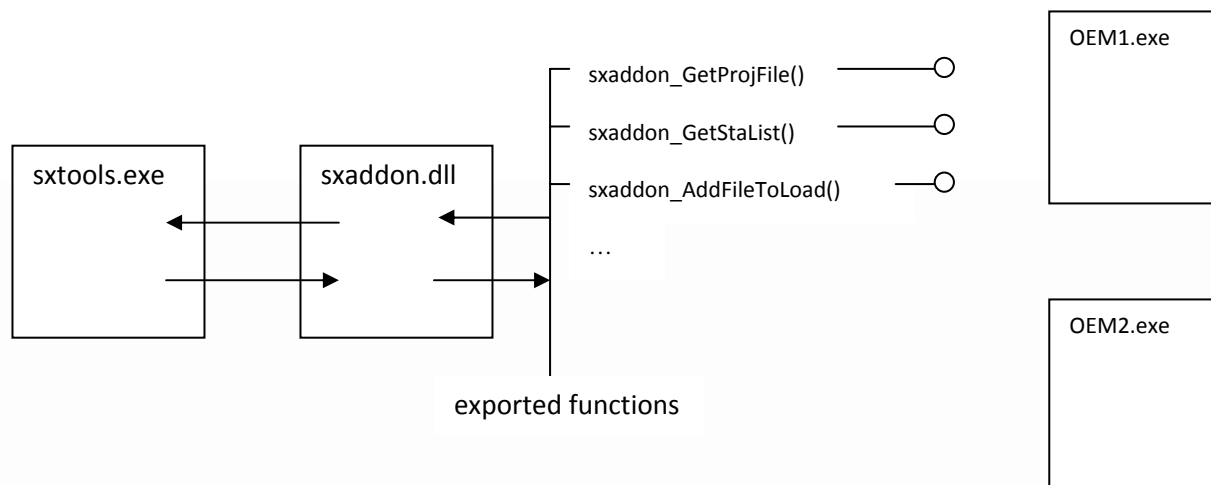
Note: The information in this file is applicable for I/O Tool Kit version 2.2 or later. Version 2.7 and later is required for use with the "User Software" lists.

Overview

This document details how to use the `sxaddon.dll` to interface with the I/O Tool Kit (`sxtools.exe`). It provides information on how to obtain, and set information such as:

1. List of IPm stations (and other stations that accept Linux applications) in the current project.
2. Name of current project.
3. List of "Files to Load" or "User Software" for a given station, or the global "User Software" list.
4. Add/Remove entries in the "Files to Load" or "User Software" lists.
5. To manipulate the "User Software" lists, include "-t=user" or "-t=globaluser" in the options strings of the applicable functions (see below for details).

Note: It is recommended that application software files be listed on the "User Software" list, so that your programs can/will be included when the controller is initialized. Include project-specific configuration files on the "Files to Load" list, so that they will be included each time a configuration load is performed.



Block diagram: `sxaddon.dll` provides a set of exported functions for OEM applications to interface with `sxtools.exe`. All interface operations between `sxtools.exe` and `sxaddon.dll` are handled automatically; users only need to call the exported functions from their applications.

Using sxaddon.dll in C/C++ programs:

Requirements:

I/O Tool Kit: version 2.2 or higher. 2.7 for “User Software” lists.

Header: sxaddonexp.h

Library: sxaddon.dll (sxaddon.lib for static linking)

The following files are provided to access the exported functions of the sxaddon.dll:

1. sxaddonexp.h (provides function prototypes)
2. sxaddon.lib (for applications that wish to statically link to the dll)

It is important to know that the sxaddon.dll can only interact with the data known to sxtools.exe. When a user opens a configuration window for a station, a copy of the station’s information is made. This copy cannot be accessed by sxtools.exe until the user returns from the configuration window by choosing OK, Cancel,... Once the user exits the configuration window the station’s information is loaded back into sxtools.exe. For this reason, external applications should only be used in a “modal” state – meaning the user should enter it and leave before going on to do other operations with sxtools.exe. Safe operation will naturally occur if the external configuration utility is called from the I/O Tool Kit. (This also makes the external configuration utility appear to be integrated into the I/O Tool Kit.) Please refer to the separately supplied documentation on adding commands to the I/O Tool Kit’s “Tools” menu.

The following functions are provided in the sxaddon.dll for public use:

sxaddon_GetStaList()
sxaddon_GetProjectFile()
sxaddon_AddFileToLoad()
sxaddon_RemoveFileToLoad()
sxaddon_GetFilesToLoadList()
sxaddon_FreeData()
sxaddon_ReadTagList()
sxaddon_FreeTagList()
sxaddon_SetStaParam()

Function descriptions:

BOOL sxaddon_GetStaList(int *pNumStations, SXADDON_STAINFO **pStaInfo, int nStaTypes, LPCTSTR lpszOptions)

inputs:

pNumStations : Provide a pointer to an int that will receive the number of stations found.

pStaInfo : Provide a pointer to a pointer of an array of SXADDON_STAINFO structures. This pointer will be allocated by the dll, and must be de-allocated when the application is done using the data. To free the data, use sxaddon_FreeData(pStaInfo).

nStaTypes : Provide the station types to add to the list. The *sxaddonexp.h* lists #define statements for the station type options. Only STATION_TYPE_IPM is implemented at this time.

lpszOptions : Not used at this time.

returns: TRUE if successful, FALSE if the list could not be obtained.

example:

```
void CSample_sxaddonDlg::OnGetStaList()
{
    m_List.DeleteAllItems();
    int nNumStations = 0;
    //Provide a pointer that the dll will
    //fill in with the address of the station list data:
    SXADDON_STAINFO *pStaInfo = NULL;
    //Ask the sxaddon.dll for the list of stations:
    BOOL bOK = sxaddon_GetStaList(&nNumStations,&pStaInfo,STATION_TYPE_IPM,"None");
    //If the call succeeded:
    if (bOK){
        //Loop through the number of stations found,
        //and add them into the dialog's list control:
        CString szVal;
        for (int x = 0; x < nNumStations;x++){
            CString szStaName = (pStaInfo + x)->szStaName;
            m_List.InsertItem(x,szStaName);
            m_List.SetItemText(x,PART_NUM_COL,
                (pStaInfo + x)->szStaPartNum);
            szVal.Format("%d",(pStaInfo + x)->nStaNum);
            m_List.SetItemText(x,STA_NUM_COL,szVal);
            szVal.Format("%lu",(pStaInfo + x)->dwSerNum);
            m_List.SetItemText(x,SER_NUM_COL,szVal);
            m_List.SetItemText(x,IP_ADDR1_COL,
                (pStaInfo + x)->szStaIP1);
            m_List.SetItemText(x,IP_ADDR2_COL,
                (pStaInfo + x)->szStaIP2);
            szVal.Format("%d",(pStaInfo + x)->nStaType);
            m_List.SetItemText(x,TYPE_COL,szVal);
        }
        //Free memory allocated during the above dll call:
        sxaddon_FreeData(pStaInfo);
    }
}
```

```
        else{
            //failed to get the station list:
            MessageBox("Failed to read station list.",
                "Error",MB_OK | MB_ICONEXCLAMATION);
        }
    }
```

BOOL **sxaddon_GetProjectFile**(TCHAR *szProjFile, int nMaxSize)

inputs:

szProjFile : Provide a pointer to a TCHAR array of nMaxSize size.

nMaxSize: Indicates the maximum size of the buffer – szProjFile.

returns: TRUE if successful, FALSE if sxtools.exe is not currently running. If sxtools.exe is running, but no project is currently open, this function will return TRUE with an empty szProjFile.

example:

```
TCHAR szProjFile[_MAX_PATH];
if (sxaddon_GetProjectFile(szProjFile,_MAX_PATH))
    SetDlgItemText(IDC_PROJ_FILE,szProjFile);
else{
    MessageBox("Failed to get project file. ","Error",MB_OK);
    SetDlgItemText(IDC_PROJ_FILE, " ");
}
```

BOOL **sxaddon_AddFileToLoad**(LPCTSTR lpszStaName, LPCTSTR lpszSrc, LPCTSTR lpszDest,
int nInsertAt = -1, LPCTSTR lpszOptions = "")

inputs:

lpszStaName: name of station being modified.

lpszSrc: Source file to add (may contain a wildcard).

lpszDest: Destination name to copy as (may contain a pattern matching expression).

nInsertAt: Zero based index to place the new entry in. -1 = add to the end of the list.

lpszOptions: Set to "-t=user" for manipulating the "User Software" list for a given station or "-t=globaluser" for manipulating the Global User Software list. When using the Global User Software list, the station name should be empty/null ("").

returns: TRUE if successful, FALSE if the operation could not be performed.

remarks:

lpszSrc can contain a wildcard "*" character. If the wildcard is used, a "*" character should be used in the *lpszDest* as well. The meaning of the "*" character in the *lpszDest* is different from the meaning in *lpszSrc*. When used in *lpszDest*, the "*" character is used as a "pattern matching" specifier.

example:

lpszSrc	lpszDest

C:\project1\ulb.*	/usr/local/bin/*

If the following files are in c:\project1:

ulb.file1.txt
ulb.file2.txt
ulb.file3.txt

The following files would get loaded as:

source	destination

c:\project1\ulb.file1.txt	/usr/local/bin/file1.txt
c:\project1\ulb.file2.txt	/usr/local/bin/file2.txt
c:\project1\ulb.file3.txt	/usr/local/bin/file3.txt

example:

```
BOOL bOK =  
sxaddon_AddFileToLoad("Station A", "c:\project1\ulb.*", "/usr/local/bin/*");
```

```
BOOL sxaddon_RemoveFileToLoad(LPCTSTR lpszStaName, LPCTSTR lpszSrc,  
LPCTSTR lpszDest, LPCTSTR lpszOptions = "")
```

inputs:

lpszStaName: Name of station to be modified.

lpszSrc: Source file of the entry to be removed. Not case sensitive.

lpszDest: Destination file of the entry to be removed. Not case sensitive.

lpszOptions: Set to "-t=user" for manipulating the "User Software" list for a given station or "-t=globaluser" for manipulating the Global User Software list. When using the Global User Software list, the station name should be empty/null ("").

returns: TRUE if successful, FALSE if the entry could not be removed, or if the entry does not exist.

remarks:

To remove an entry from the list, use this function and specify both the source and destination file names. Once removed, the list will shrink by one entry, and the list will be contiguous.

example:

```
BOOL bOK = sxaddon_RemoveFileToLoad("StationA", "c:\project1\ulb.*", "/usr/local/bin/*");
```

```
BOOL sxaddon_GetFilesToLoadList(LPCTSTR lpszStaName, int *pNumEntries,  
                                SXADDON_FILESTOLOADENTRY **pFiles,  
                                LPCTSTR lpszOptions = "")
```

inputs:

lpszStaName: Name of station to be modified.

pNumEntries: Pointer to an integer that will receive the number of entries found.

pFiles: Pointer to a pointer of type SXADDON_FILESTOLOADENTRY. The sxaddon.dll will allocate memory for the list and set this variable to point to the beginning of the allocated memory.

lpszOptions: Set to "-t=user" for manipulating the "User Software" list for a given station or "-t=globaluser" for manipulating the Global User Software list. When using the Global User Software list, the station name should be empty/null ("").

returns: TRUE if successful, FALSE if sxtools.exe is not running, or the station name could not be found.

remarks:

This function will build an array of SXADDON_FILESTOLOADENTRY structures. The sxaddon_FreeData() function must be used to free the allocated memory once the calling routine is finished using the data.

example:

```
int nNumEntries = 0;  
//Provide a pointer that the dll will  
//fill in with the address of the list data:  
SXADDON_FILESTOLOADENTRY *pStaInfo = NULL;  
//Ask the sxaddon.dll for the list of stations:  
bOK = sxaddon_GetFilesToLoadList(szStaName,&nNumEntries,&pStaInfo);  
if (bOK){  
    //Loop through the number of entries found,  
    //and add them into the dialog's list control:  
    CString szVal,szPos;  
    for (int x = 0; x < nNumEntries;x++){  
        m_FilesList.InsertItem(x,(pStaInfo + x)->szSrc);  
        m_FilesList.SetItemText(x,DEST_COL,(pStaInfo + x)->szDest);  
        szPos.Format("%d",x);  
        m_InsertAtCombo.AddString(szPos);  
    }  
    if (nNumEntries > 0){  
        m_InsertAtCombo.EnableWindow(TRUE);  
        szPos.Format("%d",x);  
        m_InsertAtCombo.AddString(szPos);  
        m_InsertAtCombo.SetCurSel(m_InsertAtCombo.GetCount() - 1);  
    }  
    else
```

```
        m_InsertAtCombo.EnableWindow(FALSE);  
        //Free memory allocated during the above dll call:  
        sxaddon_FreeData(pStaInfo);  
    }  
    else{  
        MessageBox("Failed to read files to load list.", "Error",  
            MB_OK | MB_ICONEXCLAMATION);  
    }
```

```
void sxaddon_FreeData(LPVOID pData)
```

inputs:

pData: Pointer to the block of memory to be freed.

returns: Void.

remarks:

Use this function when previous sxaddon functions have been used to allocate blocks of memory. Functions like sxaddon_GetFilesToLoadList(), and sxaddon_GetStaList() allocate arrays of structures and set a pointer to indicate the location of the allocated memory. Use this pointer to pass to sxaddon_FreeData() to free the memory previously allocated.

example:

```
//This pointer will be filled in by sxaddon.dll:  
SXADDON_FILESTOLOADENTRY *pStaInfo = NULL;  
int nNumStations = 0;  
BOOL bOK = sxaddon_GetStaList(&nNumStations, &pStaInfo, STATION_TYPE_IPM, "None");  
//If the call succeeded:  
if (bOK){  
    //Loop through the number of stations found,  
    //and add them into the dialog's list control:  
    CString szVal;  
    for (int x = 0; x < nNumStations; x++){  
        //fill in a list control with the data...  
    }  
    //Free memory allocated during the above dll call:  
    sxaddon_FreeData(pStaInfo);  
}
```

```
int sxaddon_ReadTagList(SXADDON_READTAGS *pReadInfo, SXADDON_TAGINFO **pTagData)
```

inputs:

pReadInfo: A pointer to a SXADDON_READTAGS structure used to indicate which project file, station, ... to read tags from.

pTagData: A pointer to a pointer of a SXADDON_TAGINFO structure used to receive the data.

returns: The number of tags actually read if successful. Returns <0 if not successful.

remarks:

This function allows applications to read tag information from a project file. If the file specified is the currently opened project file, the current/unsaved data will be read (a save operation from the I/O Tool Kit is not needed to get the most current information). `sxaddon_FreeTagList()` must be called to free the memory allocated by the `sxaddon.dll` once the data is no longer needed by the calling application.

SXADDON_READTAGS

```
TCHAR szStaName[80];           // station name
TCHAR szProject[_MAX_PATH];    // project file to read from
TCHAR szOptions[256];         // options (not currently used)
HWND hWndUser;                // hWnd of calling window
```

*Note: the `hWndUser` is required in order to properly maintain the project database. This value can be the handle to any window within the calling application. It must be unique (and will be if retrieved from the operating system).

SXADDON_TAGINFO

```
TCHAR szTagName[40];          // tag name
DWORD dwRegNum;               // tag register number
TCHAR szType[10];             // I/O type ("AX","AY","X","Y","LX","LY",...)
TCHAR szEngUnits[8];          // engineering units
BYTE byDecimals;              // scaled display decimal places to use
float fScaledMin;              // minimum scaled value
float fScaledMax;              // maximum scaled value
float fRawMin;                 // minimum unscaled value
float fRawMax;                 // maximum unscaled value
TCHAR szFormat[10];           // display format ("%f","%lu","%d",...)
TCHAR szOffMsg[20];           // msg to display when discrete is off
TCHAR szOnMsg[20];            // msg to display when discrete is On
TCHAR szDescription[40];       // user tag descriptor
TCHAR szModuleName[40];       // name of I/O module the tag belongs to
TCHAR szModType[80];           // part number of I/O module
TCHAR szSpecial[80];           // special feature or additional info field
```


example:

```
CString szProjFile = "c:\\sixnet tools\\projects\\test.6pj";
CString szStaName = "station1";
SXADDON_READTAGS read;
strcpy(read.szProject,szProjFile);
strcpy(read.szStaName,szStaName);
CTypedPtrArray<CPtrArray,SXADDON_TAGINFO*> tags;
SXADDON_TAGINFO *pTagInfo = NULL;
int nNumTags = sxaddon_ReadTagList(&read,&pTagInfo);
if (nNumTags > 0){
    for (int x = 0;x < nNumTags;x++){
        SXADDON_TAGINFO *pInfo = (pTagInfo + x);
        tags.Add(pInfo);
        if (pInfo != NULL){
            CString szTagName = pInfo->szTagName;
            CString szModName = pInfo->szModuleName;
        }
    }
}
//create a dialog with a list of tags found:
CTagDlg dlg(this,&tags);
dlg.DoModal();
//free the allocated memory
sxaddon_FreeTagList();
```

```
void sxaddon_FreeTagList()
```

remarks:

This function will free previously allocated tag list information from a previous call to `sxaddon_ReadTagList()`. If the calling application needs to keep it's copy of the tag data, the application should call `sxaddon_FreeTagList()` after making the copy.

example:

See the example for `sxaddon_ReadTagList()`.

```
int sxaddon_SetStaParam(SXADDON_SETSTAPARAM *pHdr, LPCTSTR lpszOptions)
```

inputs:

pHdr: pointer to a `SXADDON_SETSTAPARAM` structure (see below).

lpszOptions: not currently used.

returns:

0 if successful, < 0 if unable to set the specified parameters.

remarks:

This function allows external applications to change/modify station configuration settings. The only setting that is currently available is the protocol for a given serial port on an IPm station. Future versions may handle more settings.

SXADDON_SETSTAPARAM

```
TCHAR    szStaName[80]    // name of station that will be modified
DWORD    dwParamToSet;    // type of info. being modified
TCHAR    szParamVal[512]; // string value to set for the param.
TCHAR    szPortName[80];  // if changing port params, set this
                                // to the name of the port to be changed.

TCHAR    szOptions[128];  // future options
DWORD    dwParamVal;      // numerical value to set
DWORD    dwFlags;         // set these to appropriate string/numeric param
```

The dwParamToSet member specifies the stations configuration option that you wish to set. The only legal value at this time is SXADDON_SETPARAM_PORTPROTOCOL (see sxaddonexp.h). When using this specifier, the szPortName must be either the port's internal name (/dev/ttyS*) or the name shown on the station's label (Port A, Port B,...). The szParamVal member must be filled in with the protocol name to be used by the port. This protocol name must exist as an option for that port (factory assigned protocols or user added protocols), otherwise a failure will be returned.

The dwFlags member specifies which of the SXADDON_SETSTAPARAM members to use as the value to be set. String values can be set using the szParamVal member and numerical values are set using the dwParamVal member. When changing port protocols, the szParamVal must be used.

Legal values for dwFlags:

SXADDON_STRINGVAL	specifies that the szParamVal member is to be used
SXADDON_NUMERICALVAL	specifies that the dwParamVal member is to be used

Each type of setting may require the use of SXADDON_STRINGVAL or SXADDON_NUMERICALVAL or possibly both. The only currently available type of setting is the SXADDON_SETPARAM_PORTPROTOCOL which requires dwFlags = SXADDON_STRINGVAL, and the szParamVal to be set to the name of the protocol to be set for the given port.

example:

```
SXADDON_SETSTAPARAM hdr = {0};
strcpy(hdr.szStaName,_T("station1"));
hdr.dwParamToSet = SXADDON_SETPARAM_PORTPROTOCOL;
strcpy(hdr.szPortName,_T("/dev/ttyS0"));
strcpy(hdr.szParamVal,_T("oemprotocol_b"));
hdr.dwFlags = SXADDON_STRINGVAL;
int nResult = sxaddon_SetStaParam(&hdr);
```