SkyConductor

SOFTWARE FOR PROFESSIONAL SHOWDESIGN

Ignition plugin development guide

(Interface specification version 2)

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

Preface

SkyConductor controls external ignition systems by loading optional ignition plugins. These plugins are simple DLLs or COM-DLLs. They have to support the interface described on the following pages.

DLL registration

DLLs have to be registered with SkyConductor in order to be loaded successfully. To accomplish this, you can either write a setup file or include the following string in or at the end of your DLL: “IGNITIONSYSTEM:<Name of Plugin>=<COM-Name>”. The name of the plugin is the description that is going to be displayed in the “Choose ignition system”-Dialogue.
Simply name your compiled DLL to “something.scp” and you will be able to install it using SkyConductor. On installation, the DLLs “DLLRegisterServer”-Function will be called automatically depending on whether you supplied a COM- or normal DLL.
Non-COM-DLLs will need to have the COM-Name in the string discussed above set to “dllencap.<unique name>”. The prefix “dllencap.” makes SkyConductor treat the file as normal DLL file instead of a COM-DLL. The DLLRegisterServer-function will not be called on installation, instead the “StartIgnitionSystem”-call will be performed in order to test if the DLL is working.
After the DLLs have been successfully installed, SkyConductor automatically adds the files to the list of installed ignition systems.

Datatypes

COM-DLLs have to use different data types than normal C-DLLs can use:

Name in this document	COM-Datatype	Visual Basic Classic	VC++-Datatype*	Description
string	VT_BSTR	String	LPSTR, LPWSTR	32-bit pointer to a null-terminated string array. The COM-Variation uses UNICODE, normal DLLs use UNICODE or ANSI as defined by the GetCharSet-Function.
int32	VT_I4	Long	int	Signed 32-bit value, negative if top bit eq 1
bool	VT_BOOL	Boolean	int	FALSE = 0, TRUE > 0
void	-	(Sub)	void	No parameter or return value
HWND	VT_I4	Long	HWND	Window handle (used in the WIN32-API)

* The VC++-Datatypes MUST only be used in non-COM-DLLs.

Calling convention used

For all functions specified in this document, the STDCALL-calling convention must be used.

Exported functions

COM- and non-COM-DLLs export nearly the same functions. Whenever things differ between COM- and NON-COM-DLLs, this is noted in italics above the function description.

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

int32 GetDLLCharset (void);

- This function will only be called in the native DLL version of plugins -
Return the string format used in all other DLL functions. Return 0 to use ANSI calling, 1 to use UNICODE calling format.

int32 StartIgnitionSystem (void);

This function is called by SkyConductor right after a plugin is loaded. Return 0 to signal success, any other value to signal failure. The plugin will be unloaded on failure; the TerminateIgnitionSystem-function is not called again in this case.

void TerminateIgnitionSystem (void);

This function is called right before SkyConductor unloads the ignition system. You should use this function to perform any cleanup operations.

void ClearCues (void);

Clear internal list of transmitted cues. This is used to signal that new cues are going to be transmitted as the next step.

void AddCue (int32 address, BSTR cuetext, int32 cuetime, int32 prio, int32 flags);

Add a cue to the list of ignitions. This function is called again for every ignition to perform when shooting. Parameters are as follows:

address	32-bit ignition address. Format is specified by the ignition plugin
cuetext	Descriptive text of the ignition, added by SkyConductor
cuetime	32-bit time-offset in milliseconds. Specifies the offset to t=0 (i.e. the beginning of the show)
prio	Specifies a safety group for the cue
Flags	0 = automatic cue. Cue should fire when show time reaches cuetime 1 = manual cue. User must perform a keypress or similar to fire this cue 2 = semi-automatic cue. Cue should fire in sequence to a previous manual cue. delta(t) is calculated by substracting the current cue time from the previous manual cue time.

This function is no longer used with SkyConductor as it is replaced by the AddCueEx-function. It should be included in plugins for compatibility reasons anyway.

void CuesComplete (void);

This function is called to signal that all cues have been transmitted by the AddCue- or AddCueEx-functions.

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

void ProgramFieldSystem (void);

The plugin should now try to upload the previously added cues to the ignition systems control unit. It is left to the plugin to signal failure or other messages to the program user.

bool CheckShow (void);

Check previously added cues for plausibility. Things like time intervals between ignitions and so on should be considered here. Return TRUE if everything is ok, FALSE if the show cannot be shot because of errors found in the cue list.
This function is no longer used with SkyConductor as it is replaced by the CheckShowEx-function. It should be included in plugins for compatibility reasons anyway.

int32 GetCueStatus (int32 address);

Should measure the specified ignition circuit physically and return the result of the measurement. Function results are as follows:

Value	Meaning
>= 0	Returned value is result of measurement in Ohms * 10 (i.e. 1.5 Ohms => return value 15)
-1	Measurement error. Indicates hardware problems, missing ignition system link, ..
-2	Resistance is low enough to ignite electrical matches connected
-3	Resistance is relatively high but still okay
-4	Resistance is too high to ensure safe ignition of ematches connected

bool ShootCue (int32 address);

Shoot the ignition channel specified by the caller. Return TRUE on success, FALSE on failure.

bool ArmSystem (void);

Arm ignition system. Return TRUE on success, FALSE on failure. It is critical that you can only shoot cues after arming the system with this function! You have to implement extra safety measures to be on the safe side with this!

bool DisarmSystem (void);

Disarm ignition system. Return TRUE und success (system disarmed), FALSE on failure (system still armed). This function is even more critical than the ArmSystem-function. If you cannot verify that the ignition system is disarmed, return FALSE just to be on the safe side! SkyConductor will alert the user if this happens.

string GetCueTextFromChannel (int32 address);

Convert the address specified into a user-readable string, i.e. “1/31”. The format of the returned string depends on the ignition system. Please be aware that the address is a signed 32-bit value where negative values represent invalid addresses. Return something like “invalid address” to alert the user to this circumstance.

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

int32 GetCueChannelFromText (string addr_text);

Convert the string ignition address specified into a 32-bit value which is used by SkyConductor to represent addresses internally. Please be aware that you should only use 31 bits of the 32-bit wide variable as this is a signed int where in this case negative values represent failure (= invalid address). You should support a variety of seperation chars in this function to make it compatible to various other program functions. The best method for this is to scan the specified adress string for invalid characters normally not part of an address and convert them into valid separation chars before splitting the address into pieces and converting it.

void ShowSettingsDialog (void);

Called by SkyConductor to display the plugins setup dialog. You should use a modal form for this in order to stop the user working with SkyConductor while the ignition system setup dialoge is open.

string GetCapability (string requiredinfo);

This function is called to retrieve any information about the ignition system or the plugin itself required by SkyConductor. A table of valid arguments and their return values is specified below. The function should not distinguish between upper, lower and mixed case.

Argument passed to function	Intendet purpose / return value
HOST	Return constant string “SkyConductor”
EXTERNALMODE	Return “1” if export to external master unit is supported by plugin/ignition system, “0” if not supported.
TESTCUES	Return “1” if measuring ignition circuits via SkyConductor is supported by ignition system, “0” otherwise.
DIRECTMODE	Return “1” if shooting directly is supported by ignition system, “0” if not supported.
VERSION	Return plugin version. Descriptive use only.
IGNITIONSYSTEM	Return name(s) of supported ignition system(s). Descriptive use only.
INFOSTRING	Used to return information like copyright and disclaimer to the user. Descriptive use only.
MINIGNITIONTIME	Return minimal ignition interval to user. The unit for this is milliseconds.
CUESPERMODULE	Return the number of connectors per module if fixed. If not fixed, use the maximum number of cues available per module.
FIRSTMODULECONNECTOR	Return first module connector address, i.e. “0”, “1” or such.
ADDRESSSEPARATOR	Separation char commonly used and returned by the GetCueTextFromChannel-Function to separate module and connector address parts. Used to determine which addresses belong to the same module connection plan.
INTERFACEVERSION	Return version of used interface specification (found next to the document title). Only functions up to this spec are going to be called by SkyConductor.

void SetOwnerHWND (HWND handle);

Used to submit the handle of the SkyConductor main window to the plugin. This handle can be used to show a plugin window on top of the main windows as a toolbox or for similar purposes.

void TimeMsReached (int32 time);

Called frequently during show runtime to submit the current show time to the plugin. This can be used to synchronize an external firing system to the program when playing the music directly from the computer.

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

void StartShow (void);

Called when the show is played or resumed. The ignition system should only start shooting after receiving a TimeMsReached-call to ensure it is firing the correct cue.

void PauseShow (void);

Used to signal that the user has paused the show. The ignition system should stop firing until it receives a StartShow-call followed by a TimeMsReached-call.

void StopShow (void);

The show has been stopped (= reset) by the user. Stop shooting cues right away.

Interface specifications added by version 2

void AddCueEx (string cueinformation);

Same as AddCue-function but specifies more information about the cue. The given data is serialized using the SerializeString-function specified lateron. The CONCAT operator specified below concatenates two strings (“STRINGA” CONCAT “STRINGB” becoming “STRINGASTRINGB”):

Data format:

```
cueinformation = SerializeString(
        SerializeString(“<FIELD>=<VALUE>”) CONCAT
        SerializeString(“<FIELD>=<VALUE>”) CONCAT
        ...
);
```

The list of currently defined fields below can be extended at any time, plugins should ignore unknown field names and skip to the next field in the sequence. All field names and values are separated by a equals-sign, there can only be one value per field. A field can also contain multiple serialized strings.

Field name	Description	Value type
UID	Unique identification of each cue, important for use with the CheckShowEx-function described next.	string
NO	Number of the cue specified in the script. Any character can be used. This should only have descriptive value for the user.	string
IGN	Ignition time of this cue. Specified in milliseconds.	4 byte value; MSB first
EXPL	Burst time of this cue. Specified in milliseconds.	4 byte value; MSB first
BOUT	Time this cue is burned out. Specified in milliseconds.	4 byte value; MSB first
MODE	Describes the ignition mode of this cue.	1 byte long string. Values as follows: “A” = automatic “M” = manual “S” = semi-automatic “O” = off, don’t ignite
IGNS	Contains a list of parameters specific for each position.	For each position another serializedstring, each containing multiple serialstrings with data. Format: SerializeString(

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

		SerializeString(“<FIELD>=<VALUE>”) CONCAT SerializeString(“<FIELD>=<VALUE>”) ...);
> CHNL	Included in IGNS-Section, specifies ignition address for this position	4 byte value; MSB first
> POS	Included in IGNS-Section, specified ignition position name.	string
> ANGLE	Included in IGNS-Section, specified firing angle for this position.	int32, bit field with bits as follows: bit 0 = angle 15° left set/not set, .. bit 15 = angle 90° set/not set .. bit 30 = angle 15° right set/not set
> RACKID	Included in IGNS-Section, specified rack id for this position (internal value consisting of “<caliber>#<rack segment number>#<position name>”).	string
PRIO	Includes cue priority group/safety group	1 byte value (0-255)
SEQ	Sequence specified for the cue.	string
CNT	Effect quantity specified for the cue.	4 byte value; MSB first
DATA	Includes additional cue data.	Built by the following pseudocode: SerializeString(SerializeString(“<FIELD>=<VALUE>”) CONCAT SerializeString(“<FIELD>=<VALUE>”) ...) Field names and values are user defined. These are the same fields and values that can be set up as user defined columns in the showscript.

The order of these fields within the cueinformation parameter is fixed; if you don’t need the values specified further back, you can simply skip that part of the information and return to the caller.
The UID given as first value is important for the CheckShow-function described next.

string CheckShowEx (void);

Returns a sequence of strings made as follows:
SerializeString(
SerializeString(“<UID>=<Error text>”) CONCAT
SerializeString(“<UID>=<Error text>”) CONCAT
...
)
The UIDs included refer to the erroneous cues as transmitted by the AddCueEx-function. These UIDs are used to navigate to corresponding cues within the showscript by simply clicking on the error message. If the error is specific for a certain cue position, simply convert the position index of the position that the error pertains to into the corresponding ASCII-character and append it to the UID, prefixed with a colon (i.e. “<UID>:<1 byte position index>=<Error message>”. You can also omit the UID and the equals sign to specify any general error messages to be displayed to the user.
In case no error is found, this function must return an empty serialized string (SerializeString(“”)).

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

string GetExtensionMenus (void);

Returns a string specifying additional menus that should be appended to the “ignition system” menu in the SkyConductor main window. The string is formatted as follows:

SerializeString(
    SerializeString(“<ID>=<CAPTION>”) CONCAT
    SerializeString(“<ID>=<CAPTION>”) CONCAT
    ...
);

The ID is used to notify the plugin in case the user has clicked the menu. It may not contain any equal-signs. The caption is the text displayed as the menu entry. It may be prefixed by a “#” to indicate that the menu item should be checked. If you just specify a “-“ as the menu caption, you can create a menu separator line. In this case, no ID is needed as a separator line can’t be clicked anyways.
The menu is refreshed every time after the plugin “callmenu” notifier ist called so you can adjust the menu values according to the current plugin status.
Please be aware that the menu strings supplied with this method are not translated to the users language by SkyConductor, you have to implement your own mechanism to do that if needed.

void CallMenu (string menuID);

Called whenever the user has clicked a plugin customized menu. The menuID string specified refers to the ID given when creating the menu items with the GetExtensionMenus-function. The program window is disabled before this function is envoked and re-enabled after the function returns so the action you take with this should be rather short.

SerializedString-Storage

Serialization data is a method to embed hierarchical Structures into normal strings. Two algorythms are used by SkyConductor to serialize or de-serialize data.
The format used is described in the following table. Each serialized element is made up the following way:

“<Length of data><data>”

The data attribute contains the data stored by the element; it can also contain other serialized-data structures and be itself a member of a serialized-structure. The length attribute is used to store the length of the data section so that the original data can be recovered from the string.
The length of the length-section is not fixed but can be determined by looking at each single byte. If the topmost bit is set (0x80), more length data is following it; if the top bit is cleared, the next byte will be the first byte of the data. Because the top-bit is used as a signaling mechanism, only the 7 lower bits can be used for the value of the length-field itself. Values requiring more than one byte of length data stored this way are saved with the least significant byte (or 7 bits) first. The way this can be encoded/decoded is also shown in the pseudocode below.

Examples:

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

Length value	Stored string in hex:
13	0x0D
127	0x7F
128	0x80 0x01
253	0xFD 0x01
1034	0x8A 0x08

Encoding serialized strings

The following pseudocode encapsulates an arbitrary string:

```
Function SerializeString(EncapString)
{
	len = LengthOfString(EncapString)

	Loop until len == 0
	{
		If (len > 0x80)
		{
			APPEND to lstring: AsciiCharFromNumber( ( len BITAND 0x7F ) OR 0x80 )
			len = len SHIFTRIGHTBY 7
		}
		else
		{
			APPEND to lstring: AsciiCharFromNumber( len )
			len = 0
		}
	}
	RETURN lstring APPENDED BY EncapString
}
```

Note: In some languages a right shift is done by dividing the value by (2 ^ n) and omiting the rest.

Decoding serialized strings

The following pseudocode gets back the original string from the encapsulated/serialized string:

```
Function DeSerializeString(SerString, StartPosition)
{
	CurrentPosition = StartPosition
	lTmp = 0xFF
	lShift = 0
	Loop until (len BITAND 0x80) > 0
	{
		lTmp = AsciiValueOfChar( CharOfString ( SerString, CurrentPosition ) )
		len = len BITOR ( ( lTmp BITAND 0x7F ) SHIFTLEFTBY lShift )
		lShift = lShift + 7
		INCREMENT CurrentPosition BY 1
```

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de

```
        }
        (StartPosition = CurrentPosition + len)
        RETURN ALL CHARS IN SerString FROM CurrentPosition TO StartingPosition + len
}
```

Notes:

- In some programming languages a left shift is done by multiplying the value by (2 ^ n), n being the number of bits to shift to the left.
- Depending on your programming language you have to call the decoding function with CurrentPosition = 1 or = 0 (meaning the first char).
- If you want to return the value of the next starting position to the called, implement the line StartPosition = CurrentPosition + len, found before the RETURN in the pseudocode. This will enable you to call the function over and over on the same string, each time returning the next serialized section.

SkyConductor Softwareentwicklung
Tobias Hagemeier
Ernststr. 27
45307 Essen

Telefon: +49-(0)201 / 46 979 48
Fax: +49-(0)201 / 46 979 49
Email: info@skyconductor.de
Homepage: http://www.skyconductor.de