SENSORAY CO., INC.

2400
Software Development Kit
Version 1.0.1

7313 SW Tech Center Dr.
Tigard, OR 97223
Phone 503.684.8005 • Fax 503.684.8164
sales@sensoray.com
www.sensoray.com

Table of Contents

Introduction

Sensoray's 2400SDK is a software development kit that has been developed to allow OEM's to build their own applications around Sensoray's high performance MPEG-1/2 decoder technology. This SDK is also the basis for Sensoray's full featured player; our 2400DCS display and control software.

While designed to decode multiple MPEG stream simultaneously with very low latency and CPU usage, Sensoray's MPEG-1/2 decoder still maintains high image quality. It is highly optimized, using MMX/SMID for decoding and DirectX© for image display. In addition, Sensoray's robust decoder design behaves reliably when receiving MPEG streams over unreliable networks, where data can be corrupt or network packets can be lost. The decoder can also decode from files, i.e. MPEG video clips.

The 2400SDK's API (application program interface) is the interface to Sensoray's decoder and display DLL (dynamic link library). The API includes a powerful set of decoder and display functions to simplify and shorten the development cycle of sophisticated video stream player applications. The API is both directly C and C++ compatible. All the API functions are built into a standard DLL; interfacing to other languages is also possible.

A sample application, written in MFC/C++, that demonstrates the use of all the API functions is included in the 2400SDK. The sample application comes with its source code. This source code is free to use for whatever purpose and is an ideal starting point for your application.

The 2400SDK decoder is primarily intended to be used with Sensoray's Model 2412/16 network MPEG encoder system but could be used with other streaming sources. The decoder supports both Sensoray's streaming protocol as well as RTP (real-time protocol). Our RTP implementation closely follows the standard and has been tested with QuickTime's RTP.

Changes in 2412 SDK v.1.0

The current release includes the first version of SDK.

Software Installation

Copy protection

The 2400SDK includes a copy protection mechanism. This protection system will also copy protect your derived application. Any person attempting to use the DLL (sspdll.dll) beyond the 30 day trial period must obtain/purchase a license key from Sensoray Co. Inc.

VERY IMPORTANT: When the running the 2400SDK for the first time a license dialog will pop up and you will have this and only this oportunity to enable your 30 day trial period. For your 30 day trial period you must press the "Enable 30 Day Trial License". If you accidentally press one of the other buttons you will not have another opportunity to enable the trial license. This problem will be fixed in a future release.

Requirements

Minimum processor: Pentium III 600 MHz. A Pentium IV 2 GHz or faster is recommended.

Operating system: Windows NT, Windows 2000, or Windows XP. Windows 98 is not supported. Linux will be supported soon.

Minimum system RAM: 128 Mbytes. 256 Mbytes or more is recommended.

Video card: A high performance video card is highly recommended. The video card must support Microsoft DirectX and should have at least 64MB of video ram

DirectX: Version 8.1 or more recent. If you do not already have DirectX installed on your system (Windows XP installations include DirectX), you must obtain a DirectX runtime package from Microsoft. The exact version you need depends on your operating system and can be download directly from Microsoft's web site at the following URL:

http://www.microsoft.com/downloads/search.aspx?displaylang=en&categoryid=2

Read the DirectX installation instructions carefully before installing.

Installation

To install the 2400SDK run the 'windows-setup.exe' installation program included with the distribution. We recommend using the default installation settings. Before installing make sure you have met the requirements above.

VERY IMPORTANT: When the running the 2400SDK for the first time a license dialog will pop up and you will have this and only this opportunity to enable your 30 day trial period. For your 30-day trial period you must press the "Enable 30 Day Trial License".

If you accidentally press one of the other buttons, you will not have another opportunity to enable the trial license. This problem will be fixed in a future release.

Upgrading from an earlier 2400SDK version

This is the first release of 2400SDK.

Performance issues

In most cases, the performance of an application based on the 2400SDK is dependent on the amount of system memory, amount of video memory, video card performance, and CPU performance. The decoding and displaying of MPEG streams, especially multiple stream simultaneously, taxes even high performance systems. Do not attempt to decode a large number of streams on a minimal system.

We strongly recommend keeping your CPU usage below 80 – 90 % for a responsive user interface and good 2400SDK (or derived application) performance.

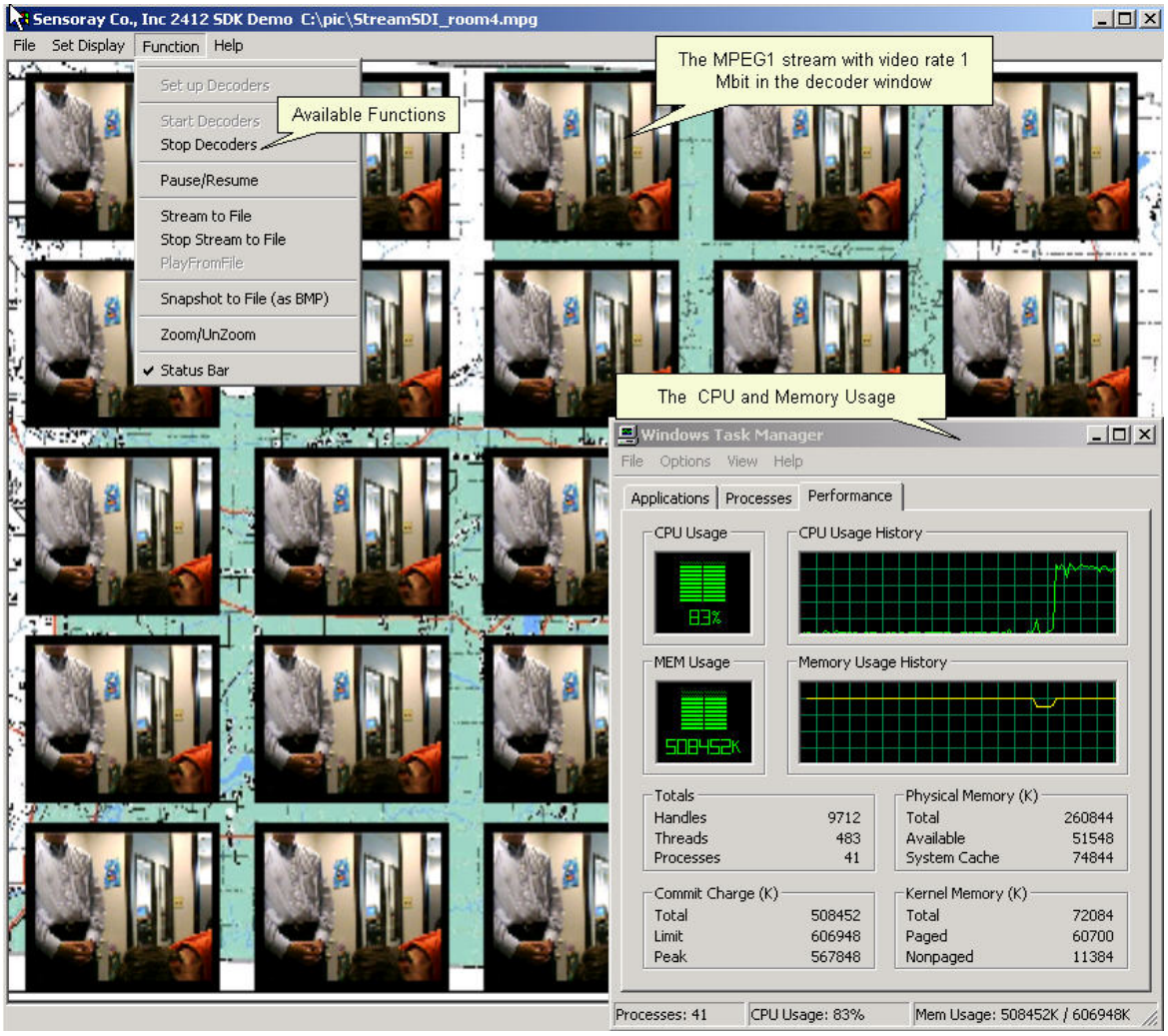
To occasionally see the CPU rise to 100% is acceptable but it should never stay at 100%. When the system stays at 100%, various MPEG decoder queues start to fill, latency increases, and the number of dropped frame becomes significant. Reliability may also be affected.

As a rough benchmark, the 2400SDK Sample program, with about 85% CPU usage, can decode, 25 MPEG-1/1Mbit video streams or 8 MPEG-2/5Mbit video streams, simultaneously on a 3GHz P4 PC with 256 Mbytes of system RAM and a GeForce4/128Mbyte video card.

2400SDK Sample Application

Description

The 2400SDK sample application is a working windows application that allows you to display in real-time multiple MPEG-1 and MPEG-2 video streams.



The 2400SDK sample application is a simple MFC (Microsoft Foundation Classes) windows application. The source code for the application is included and demonstrates the usage of the 2400SDK API (application program interface) functions. The API consists of a DLL (sspdll.dll and slsApi.dll), a header (sspdll.h) file, and a library file for MS Visual Studio projects (sspdll.lib).

Note: The pre-built sample application, sspdemo.exe, is located in the folder c:\Program Files\Sensoray\2400SDK (default installation path).

Note: The Microsoft Visual Studio project files and source code for the sample application are located in c:\Program Files\Sensoray\2400SDK\source. Remember to place a copy of both DLLs (i.e. ssp.dll and slsApi.dll) into the same directory as sspdemo.exe or into a system directory that is in the path. After a build sspdemo.exe will typically be located in c:\Program Files\Sensoray\2400SDK\source\Release or c:\Program Files\Sensoray\2400SDK\source\Debug.

Note: Sample pictures for use as background images with the demo are located in c:\Program Files\Sensoray\2400SDK\pic. Examples for playlist.txt and 'sdp' files are in c:\Program Files\Sensoray\2400SDK\setup.

2400SDK Sample Application Main Menu

File

File->Get HTTP Connection requests MPEG video stream from remote 2412 stream server using the 2412's URL for the sdp data. A typical entry would be of the form: http://10.139.9.28/stream0.sdp

File->Get SDP File obtains connection information from an sdp file: c:\myFile.sdp. An sdp file describes a stream from either a 2412 where you do not want to use its URL (see Get HTTP Connection) or a non-2412 stream server. Shown below, are typical contents of an sdp file:

```
v=0
o=- 1027789063 1027789063 IN IP4 10.9.10.11
s=Sensoray ss0 MPEG session
i=ss0_sender
t=0 0
a=tool:Sensoray2412Streamer
a=type:broadcast c=IN IP4 10.9.10.11/127
m=video 18888 RTP/AVP 32
a=framerate:30
a=picturesize:3
```

File->Load From TextList selects a file that contain a list of URLs (see Get HTTP Connection). This file is used to automate the connection to multiple 2412 streams. The contents of a typical TextList **file are shown below**:

http://10.9.9.29/stream0.sdp
http://10.10.9.11/stream0.sdp
http://10.9.9.10/stream0.sdp
http://10.9.9.29/stream1.sdp

File->Save As TextList saves the current connection information into a TextList file (.txt extension).

File->Clear Display removes all decoder connections and removes the background image. (Note: The directX display surface remains alive and you can build a new set of connections and/or use a new background image).

File->Exit

Set Display

Set Display->Create Display creates the directX display surface.

Set Display->Set Color Background is used to set the color of the background. (This can be done at any time and even when the decoders are active)

Set Display->Set Image Background is used to set an image as background (the standard image formats *.bmp,*.jpg,*.dds,*.dib,*.png,*.tga are accepted).

Functions

Functions->Setup Decoders puts the predefined connection on the display surface and creates the directX display surface.

Functions->Start Decoders starts, if it possible, all predefined decoders.

Functions->Pause/Resume pauses or resumes the video stream for network stream decoders.

Functions->Stream to File simultaneously sends the received MPEG video stream to a file (extension .mpg). This applies to the first decoder only.

Functions->Functions->Stop Stream to File stops the stream to file operation.

Functions->Play From File plays an MPEG stream video from a file.

Functions->Snapshot to File (as BMP) takes a snapshot of the currently decoded image and saves it to a Windows compatible bitmap file (BMP).

Functions->Zoom/Unzoom zooms to the predefined zoom level all streaming decoders and, if possible, all predefined decoders

Functions->Status Bar enables/disables the status bar.

Help

Help->License Manager brings up the license manager dialog box. Note: This option is available only before the creation of the directX display surface.

Help->2400SDK License Information displays the status of the license.

Note: To simplify this sample application, any function that modifies a setting modifies the same setting for each decoders.

Typical steps used to run the 2400SDK sample application

1. **Set Display->Create Display** creates the directX display surface. After creating the display surface you can change the background of the display at any time. Background pictures samples can be found in c:\ProgramFiles\Sensoray\SDK2400\pic.

2. **File->Get HTTP Connection**, or **File->Get SDP File,** or **File->Load From TextList** sets up the available or possible connections. If you do not have 2412 stream servers you can open an sdp file that describes the stream or play an MPEG-1/2 clip from a file. If you want to connect to multiple stream simultaneously open TextList file. Example of both sdp and TextList files are located in c:\ProgramFiles\Sensoray\SDK2400\Setup.

3. **Functions->Setup Decoders** this makes the decoders defined in step 2 appear on the display surface. The position of the decoders is dependent on the number of decoders.

4. **Functions->Start Decoders** starts all the decoders.

5. **Functions->Stop Decoders, Functions->Pause/Resume, Functions->Stream to File, Functions->Snapshot to File (as BMP), Functions->Zoom/Unzoom** are now available. The function **Functions->Stream to File** is available only if decoding from a network stream**.**

6. **File->Save As TextList** saves the setup of the decoders as a text file. This file can be manually edited.

Building an application with the 2400SDK

Files to be included in the project

The following files are distributed with the 2400SDK:

- `sspdll.h` – contains data types, constant definitions, exported function definition;
- `sspdll.dll` – dll library;
- `slsApi.dll` – dll library;
- `sspdll.lib` – dll library – lib file for MS Visual Studio C/C++ users;

NOTE: When building an MS Visual Studio application with `2400SDK`, it is necessary to include `sspdll.lib` in the project.

NOTE: The 2400SDK sample application was built using Microsoft's MFC for clarity. An example of nearly every function call is present in the source code.

NOTE: All modules containing 2400SDK API function calls must include the `sspsll.h`. Please refer to the sample application's source code and included MS Visual Studio's project files for an example of an application using the 2400SDK API.

Data types reference

SDPInfo structure

Syntax

```
typedef struct
{
    CHAR                ServerName[128];      (in)
    INT                 Width;                (in)
    INT                 Height;               (in)
    INT                 FrameRate;            (in)
    DECODER_SETTINGS    DecSettings;          (in)
} SDPINFO, *LPSDPINFO, *PSDPINFO;             (in)
```

The SDPInfo structure contains information about video stream obtained from an sdp file or a 2412 sdp URL.

Members

CHAR ServerName[128]
Char array contains the unique symbolic name for the stream the server.

INT Width
The Width of the frame.

INT Height
The Height of the frame.

INT FrameRate
The frame rate of the MPEG video (typically 30 fps for NTSC).

DECODER_SETTINGS DecSettings
The structure containing information about the stream.

DECODER_SETTINGS structure

Syntax

```
typedef struct
{
    // For network input
    INT         DecoderType;                (in)
    INT         Port;                       (in)
    CHAR        RemoteAddress[16];          (in)
    // For file input
    CHAR        FileName[128];              (in)
    HANDLE      FileHandle;                 (in)
    // Stream type
    INT         StreamType;                 (in)
} DECODER_SETTINGS, *LPDECODER_SETTINGS, *PDECODER_SETTINGS;SDPINFO,
*LPSDPINFO, *PSDPINFO;
```

The DECODER_SETTINGS structure contains information about network or file stream.

Members

INT DecoderType;

Decoder type required:

DEC_UCAST_UDP	– Unicast Sensoray stream decoder.
DEC_UCAST_RTP	– Unicast RTP stream decoder.
DEC_MCAST_UDP	– Multicast Sensoray stream decoder.
DEC_MCAST_RTP	– Multiicast RTP stream decoder.
DEC_TCP	– Not impemented.

INT Port;

Ip port that must used to receive the stream.

CHAR RemoteAddress[16];

Ignored (can be NULL) if DEC_UCAST_UDP or DEC_UCAST_RTP decoder. Stream server address if DEC_TCP decoder (not implemented), Multicast group address if DEC_MCAST_UDP or DEC_MCAST_RTP multicast decoder.

CHAR FileName[128]

The file name. If used set FileHandle to NULL.

HANDLE FileHandle

The file handle. If not NULL FileName is ignored.

INT StreamType

The MPEG Stream type: STREAM_TYPE_ES_AUDIO, STREAM_TYPE_ES_VIDEO, STREAM_TYPE_SYSTEM, STREAM_TYPE_TRANSPORT.

(At this time only STREAM_TYPE_ES_AUDIO is supported. The other stream types will be available soon).

DISP_SETTINGS structure

Syntax

```
typedef struct
{
    INT         PosX;                       (in)
    INT         PosY;                       (in)
    BOOL        Centered;                   (in)
    INT         DefaultScale;               (in)
    WORD        ZoomedScale;                (in)
    BOOL        ThumbNailUseFile;           (in)
    INT         ThumbNailWidth;             (in)
    INT         ThumbNailHeight;            (in)
    TCHAR       ThumbNailFile[128];         (in)
    DWORD       ThumbNailColor;             (in)
} DISP_SETTINGS, *LPDISP_SETTINGS, *PDISP_SETTINGS;
```

The DISP_SET structure contains information about display setting.

Members

INT PosX
Horizontal position of viewer

INT PosY
Vertical position of viewer

BOOL Centered
TRUE - viewer centered on position. FALSE - position is top left corner of viewer

INT DefaultScale;
Default scale: SCALE_X2, SCALE_X1, SCALE_X1_2, SCALE_X1_4, SCALE_X1_8, SCALE_X1_16

WORD ZoomedScale;
Zoomed scale: SCALE_X2, SCALE_X1, SCALE_X1_2, SCALE_X1_4, SCALE_X1_8, SCALE_X1_16

BOOL ThumbNailUseFile;
TRUE if file is to be used

INT ThumbNailWidth;
Thumbnail Width

INT ThumbNailHeight;
Thumbnail Height

TCHAR ThumbNailFile[128];
Bitmap file name for thumbnail or NULL for color.

DWORD ThumbNailColor;
Color of thumbnail if filename is NULL.

AUDIO_SETTINGS structure

Syntax

```
typedef struct
{
    INT     Volume;                          (in)
    INT     Balance;                         (in)
} AUDIO_SETTINGS, *LPAUDIO_SETTINGS, *PAUDIO_SETTINGS;
```

The AUDIO_SETTINGS structure contains information about audio stream.

Members

INT Volume
Audio volume setting.

INT Balance;
Audio volume setting.

MPEGINFO structure

Syntax

```
typedef struct
{
    INT   coded_picture_width;    // Picture width out of decoder
    INT   coded_picture_height;   // Picture height out of decoder
    INT   horizontal_size;        // Horizontal size from stream
    INT   vertical_size;          // Vertical size from stream
    INT   display_horizontal_size;// Display horizontal size from
                                  // stream
    INT   display_vertical_size;  // Display vertical size from stream
    INT   aspect_ratio;           // Aspect ratio from stream
    INT   bit_rate;               // Bit rate from stream in bits/sec
                                  // from stream
    INT   picture_rate;           // Picture rate code from stream
    INT   mpeg2;                  // Non-zero if MPEG-2
    INT   gop_count;              // Running count of GOPs processed
    INT   hour;                   // Time - hours from stream
```

```
    INT   minute;                 //    - minutes from stream
    INT   sec;                    //    - seconds from stream
    INT   frame;                  // Frame count
    DWORD level;                  // Queue level in percent
    DWORD thread_id;              // Decoder thread id
    INT   display_id;             // The decoders display id
    INT   width;                  // Width of display
    INT   height;                 // Height of display
    INT   audio_bit_rate;         // Audio bit rate
    INT   audio_channels;         // Number of audio channels
    INT   audio_compression_system; // Audio compression system.
} MPEGINFO, *LPMPEGINFO, *PMPEGINFO;
```

The MPEGINFO structure contains information MPEG stream.

License Information structures

SSP_LICENSE structure

Syntax

```
typedef struct
{
    DWORD   State;          // State of license;
    DWORD   DaysLimit;      // License length in days.
    DWORD   MaxDecoders;    // Maximum number of simultaneous decoders
                            // allowed by license.
    DWORD   DecodersUsed;   // Number of decoder that are opened at
                            // this time.
    DWORD   DaysUsed;       // Number of days used.
    LICDATE StartDate;      // Date usage started.
    LICDATE EndDate;        // Last day license is valid.
} SSP_LICENSE, *LPSSP_LICENSE, *PSSP_LICENSE;
```

The SSP_LICENSE structure contains information about license. This information is used by the function SSP_CheckLicense().

LICDATE structure

Syntax

```
typedef struct
{
    DWORD Year;
    DWORD Month;
    DWORD Day;
} LICDATE;
```

The LICDATE structure contains the date structure.

Functions reference

SSP_DisplayOpen()

The SSP_DisplayOpen() function creates the "Direct X" display with defined Width, Height, Color, and display's refresh rate

```
INT SSP_DisplayOpen(
    HWND    hWnd,                (in)
    INT     nWidth,              (in)
    INT     nHeigth,             (in)
    INT     nRefreshRate,        (in)
    DWORD   nColor,              (in)
    LPDXHAN lphDisp              (out)
);
```

Parameters

hWnd
Handle to parent window.
nWidth
Width in picture units of desired display.
nHeigth
Height in picture units of desired display.
nRefreshRate
Refresh rate of the display.
nColor
Background RGB color of display.
lphDisp
Handle to DX display.

Return values

Returns the handle to display when successful or an error code on failure. All error codes are less than 0 and a list of the error codes can be found in the DirectX developer documentation.

Note:

The size of the display that can be opened depends on the amount available system memory and video memory. The common errors for SSP_DisplayOpen() are "out of memory to allocate display" and "out of video memory to perform the operation"

SSP_DisplayClose()

The SSP_DisplayClose() function close the "Direct X" display and release all resources . If you use the status of the handle in you program – be careful – function does not set up the handle hDisp to NULL.

```
void SSP_DisplayClose(
    DXHAN   hDisp         //(in)
);
```

Parameters

hWnd

Handle to an opened display.

Return values

None.

Notes

SSP_DisplayBackgroundFile()

The SSP_DisplayBackgroundFile() function sets the background of the display to a graphics file. Supports standard graphics formats like .jpg, .bmp, .dds, .dib, .png, .tga.

```
void SSP_DisplayBackgroundFile(
    DXHAN   hDisp,          // (in)
    PTCHAR  pFileName )     // (in)
);
```

Parameters

hDisp

Handle to an opened display.

pFileName

Pointer to graphics file.

Return values

None

SSP_DisplayBackgroundColor()

The SSP_DisplayBackgroundColor() function sets the color of the display to the specified color in RGB format.

```
void SSP_DisplayBackgroundColor(
   DXHAN   hDisp,                  // (in)
   DWORD nColor  )                 // (in)
);
```

Parameters

hDisp
Handle to an opened display.

nColor
RBG color.

Return values

None

Notes

SSP_DisplayDFrameTime ()

The SSP_displayDFrameTime() function gets the time between decompressed frames in milliseconds.

```
INT SSP_DisplayDFrameTime(
   DWHAN   hDW )          // (in).
);
```

{**Parameters**

hDW
Handle to an opened display.

Return values

If the time could not be obtained returns (–1), otherwise the time between decompressed frames is returned in millisecond.

Notes

SSP_ChangeDecoderWindowDisplay ()

The SSP_ChangeDecoderWindowDisplay() function changes the display windows of an open decoder window.

```
void SSP_ChangeDecoderWindowDisplay(
    DWHAN   hDW,          // (in)
    DXHAN   hNewDisp      // (in)
);

hDW
   Handle to an opened decoder window.

hNewDisp
   Handle to the new opened display.
```

Return values

none

Notes

SSP_ IsInDecoderWindow ()

The SSP_IsInDecoderWindow () function determines whether or not the given position ((x, y) coordinates) is within a decoder display.

```
INT SSP_IsInDecoderWindow(
    DWHAN   hDW,           // (in)
    INT     x,             // (in)
    INT     y,             // (in)
    LPRECT  pRect );       // (out)
);

hDW
   Handle to an opened display window.
x
   Horizontal position.
y
   Vertical position
pRect
   The Rectangle describing decoder display window (position and size).
```

Return values

Returns 0, if the position is outside the decoder window rectangle. Otherwise, it returns a non zero value and sets pRect (the rectangle of the decoder window).

Notes

SSP_DisplayPause ()

The SSP_DisplayPause () function pauses the decoder display but not the decoder itself.

```
INT SSP_DisplayPause(
   DWHAN    hDW                 // (in)
);
```

`hDW`
Handle to an opened decoder window.

Return values

If no error occurs, returns 0. Otherwise, it returns (–1).

Notes

SSP_DisplayResume ()

The SSP_DisplayResume () function resumes the paused decoder's stream to display.

```
INT SSP_DisplayResume (
   DWHAN    hDW                 // (in)
);
```

`hDW`
Handle to an opened decoder window.

Return values

If no error occurs, returns 0. Otherwise, it returns (–1).

Notes

SSP_DisplayCaptureBMP ()

The SSP_DisplayCaptureBMP () function captures the decoded picture to bitmap file.

```
INT SSP_DisplayCaptureBMP(
   DWHAN   hDW,                    // (in)
   HANDLE  hBmpFile                // (in)
);
```

hDW
Handle to an opened decoder window
hBmpFile
Handle to an opened for write bitmap file.

Return values

If no error occurs, returns 0. Otherwise, it returns (–1).

Notes

SSP_DecoderWindowSelected ()

The SSP_DecoderWindowSelected () function puts the yellow selection ring around the selected decoder window (unselects all others).

```
void SSP_DecoderWindowSelected(
   DWHAN    hDW                    // (in)
);
```

hDW
Handle to an opened decoder window.

Return values

none

Notes

SSP_DisplayText ()

The SSP_DisplayText () function displays the text on the decoder window. The text starts in the top left corner of the decoder window.

```
void SSP_DisplayText(
   DWHAN      hDW,          // (in)
   INT        nOffsetX,     // (in)
   INT        nOffsetY,     // (in)
   PTCHAR     Text,         // (in)
   INT        TextSize,     // (in)
  INT         Color         // (in)
);
```

hDW p
Handle to an opened decoder window.
nOffsetX
Text starting position offset from left side viewer. If SAMEPOS position is not changed.
nOffsetY
Text starting position offset from top side viewer. If SAMEPOS position is not changed.
Text
Pointer to text string. If NULL, text is not changed
TextSize
Text size (height) in pixels when scale is x1. If (-1), size is not changed.
Color
24 bit RGB color value. If (-1), color is not changed.

Return values

none

Notes: Broken!

SSP_OpenDecoderWindow ()

The SSP_OpenDecoderWindow() function opens the decoder's window. The windows on the surface of Direct X display with defined position and size. The decoder window is the window for the decoded video stream.

```
DWHAN SSP_OpenDecoderWindow (
    DXHAN               hDisp,          // (in)
    AUHAN               hAudio,         // (in)
    HWND                hWnd,           // (in)
    INT                 nMessage,       // (in)
    LPDISP_SETTINGS     pDispSettings,  // (in)
    INT                 nSet            // (in)
);
```

hDisp
Handle to an opened display.
hAudio
Handle to an opened audio output. NULL if no audio.
hWnd
Handle window for receiving message every time a frame is decoded. NULL if not needed.
nMessage
Message to send. Ignored if the handle hWnd is NULL .
pDispSettings
Pointer to DISP_SETTINGS structure.
nSet
A constant that sets the picture size and type. There allowed values are:
- DEFAULT - use default position/size;
- ZOOMED - use zoomed position/size;
- THUMBNAIL- use thumbnail position/size.

Return values

If no error occurs, the HANDLE of the decoder window is returned. Otherwise, NULL is returned.

Note: Set is used to select either the DEFAULT, ZOOMED, or THUMBNAIL settings from the supplied DISP_SETTINGS structure, **pDispSettings.**

```
When nSet = DEFAULT the following setting applies:
    DispSettings.DefaultScale.
When nSet = ZOOMED the following setting applies:
    DispSettings.ZoomedScale.
When nSet = THUMBNAIL the following settings apply:
    DispSettings.ThumbNailUseFile
    DispSettings.ThumbNailWidth
    DispSettings.ThumbNailHeight
    DispSettings.ThumbNailFile
    DispSettings.ThumbNailColor
```

For DEFAULT and ZOOMED several scaling factors are available: SCALE_X2, SCALE_X1, SCALE_X1_2, SCALE_X1_4 SCALE_X1_8 SCALE_X1_16 (see sppdll.h).

SSP_CloseDecoderWindow ()

The SSP_CloseDecoderWindow () function closes the decoder window created by SSP_OpenDecoderWindow().

```
void SSP_CloseDecoderWindow(
   DWHAN   hDW                    // (in)
);
```

hDW
Handle to an opened decoder window.

Return values

none

Notes

SSP_ModifyDecoderWindow ()

The SSP_ModifyDecoderWindow () function sets the decoder's window in predefined position and size.

```
DWHAN INT SSP_ModifyDecoderWindow(
   DWHAN                hDW,            //  (in)
   LPDISP_SETTINGS      pDispSettings,  //  (in)
   INT                  nSet            // (in)
);
```

hDW
Handle to an opened decoder window.
pDispSettings
Pointer to a display settings structure.
nSet
the const pointed to the type and size of the picture. There are possible:
- DEFAULT - use default position/size;
- ZOOMED - use zoomed position/size;
- THUMBNAIL- use thumbnail position/size.

Return values

If error occurs, returns (–1). Otherwise, it returns 0.

Note: See SSP_OpenDecoderWindow() for details on nSet and pDispSettings.

SSP_StartDecoder ()

The SSP_StartDecoder () function starts the video stream to the decoder window.

```
DWHAN INT SSP_StartDecoder(
   DWHAN                    hDW,          //  (in)
   LPDECODER_SETTINGS       pDecSettings  //  (in)
);
```

hDW
Handle to an opened decoder window
pDispSettings
Pointer to a decoder setting structure.

Return values

If no error occurs, returns 0. Otherwise, it returns (-2) for exceeded license, and (-1) for others errors.

Notes

This function allows to start video stream from the file. In this case we need to set up the next parameters of the decoder setting structure:

DecSettings.DecoderType	- DEC_FILE - means – open the file as a stream.
DecSettings.FileHandle	- the handle of the opened file or NULL. In the second case the file will be opened.
DecSettings.StreamType	- STREAM_TYPE_ES_VIDEO type of the stream.
DecSettings.FileName	- the name of the file to play.

The "stream from the file" can be control the same functions, as video stream for decoder.

SSP_StopDecoder ()

The SSP_StopDecoder () function stops the video stream.

```
INT SSP_StopDecoder(
   DWHAN   hDW                  // (in)
);
```

hDW
Handle to an opened decoder window

Return values

If no error occurs, returns 0. Otherwise, it returns (-1).

Notes

SSP_StartStreamFile ()

The SSP_StartStreamFile () function starts streaming video MPEG data to a file.

```
INT INT SSP_StartStreamFile(
   DWHAN   hDW,                  // (in)
   HANDLE  hFile                 //(in)
);
```

hDW
Handle to an opened decoder window
hFile
Handle to an opened for write MPEG file

Return values

If no error occurs, returns 0. Otherwise, it returns (-1).

Notes

SSP_StopStreamFile ()

The SSP_StopStreamFile () function stops streaming video MPEG data to a file.

```
INT INT SSP_StopStreamFile(
   DWHAN   hDW                   // (in)
);
```

hDW
Handle to an opened decoder window

Return values

If no error occurs, returns 0. Otherwise, it returns (-1).

Notes

SSP_GetLastDecoderError ()

The SSP_GetLastDecoderError () function returns the last decoder's error.

```
INT SSP_GetLastDecoderError(
   DWHAN   hDW                    // (in)
);
```

hDW
Handle to an opened decoder window

Return values

If no error occurs, returns the last decoder error. Otherwise, it returns (-1).

Notes

SSP_GetDecoderSourceAddr ()

The SSP_GetDecoderSourceAddr () function gets the video stream source address.

```
void SSP_GetDecoderSourceAddr(
   DWHAN   hDW,                   //  (in)
   PTCHAR  pSourceIP              // (out)
);
```

hDW
Handle to an opened decoder window
pSourceIP
Pointer to buffer accepting returned address

Return values

Pointer to the buffer contained the source address

Notes

SSP_GetDecoderInfo ()

The SSP_GetDecoderInfo () function gets the information about the video stream as MPEGINFO structure.

```
INT SSP_GetDecoderInfo(
   DWHAN       hDW,                  //  (in)
   LPMPEGINFO  pInfo                  //  (out)
);
```

hDW
Handle to an opened decoder window
pInfo
Pointer to a MPEGINFO structure

Return values

If no error occurs, returns 0 and the pointer to the MPEGINFO structure.
Otherwise, it returns (-1).

Notes

SSP_GetSDPInfo ()

The SSP_GetSDPInfo () function starts the 2412 server and gets the SDP information about the video stream.

```
INT SSP_GetSDPInfo(,);
 (
   PTCHAR pUrl,                  //  (in)
   LPSDPINFO SDPInfo             //  (out)
);
```

pUrl
Pointer to the character buffer contained the http address of the video stream source or the name of the SDP file.

SDPInfo
Pointer to a MPEGINFO structure

Return values

If no error occurs, returns 0, otherwise the code of the error.

ERR_HTTP_SDP - Failed on HTTP connection, the main reason usually the 2412 server is down.
ERR_FILE_SDP - Failed on parsing the SDP file.
ERR_SOCKET_VERSION - Failed on http connection.
ERR_BUSY_SDP – the server is busy (for unicast mode the 2412 server can send the stream to only one receiver, for multicast mode the 2412 server will provide the stream to the many receivers).
(The return values less than 0 are errors)

Notes

The URL address is usually in the form like http://10.9.9.29/stream0.sdp.
The SDP file in the form like c:\\myfile.sdp.

License functions

SSP_CheckLicense ()

The SSP_CheckLicense () function checks the license information and fill up the license structure.

```
API INT SSP_CheckLicense(
        LPSSP_LICENSE pLicense // (out));
);
```

Parameters

pLicense
User supplied pointer to an allocated SSP_LICENSE structure.

Return values

Returns 0 in case of success, or otherwise (-1), if the license does not exist.

Notes

SSP_LicenseManager ()

The SSP_LicenseManager () function brings the license manager dialog on the top of the window.

```
INT SSP_LicenseManager(
        HWND    hWnd        // (in) Handle to parent window
);
```

Parameters

hWnd
Handle to parent window.

Return values

Returns 0 in case of success, or otherwise (-1), if the dialog is failed.

Notes

The license manager dialog is available only before creating the directX display surface.

The dialog shows the "product identification code" which corresponds the Sensoray_SSPDDL product, the "Reference code" what will use for creating the license key.
The dialog box also contain the current information about the status of the license, including the number of licensed decoders, expiration date.

Audio functions

SSP_AudioOpen ()

The SSP_AudioOpen () function opens the audio output device.

```
AUHAN SSP_AudioOpen(
   HWND    hWnd                    // (in)
);
```

Parameters

hWnd
Handle to parent window.

Return values

Returns a handle to the audio device in case of success, or otherwise (-1).

Notes: Not yet implemented!

SSP_AudioClose ()

The SSP_AudioClose() function close the audio output device.

```
AUHAN SSP_AudioClose(
   AUHAN   hAudio                  // (in)
);
```

Parameters

hAudio
Handle to an opened audio device.

Return values

Returns 0 in case of success, or otherwise (-1).

Notes: Not yet implemented!

SSP_AudioControl ()

The SSP_AudioControl() function sets volume and balance for an audio output device.

```
SSPLAYER_API INT SSP_AudioControl(
   AUHAN                hAudio,          //  (in)
   LPAUDIO_SETTINGS     pAudioSettings  // (in)
);
```

Parameters

hAudio
Handle to an opened audio device.
pAudioSettings
Pointer to audio settinf structure.

Return values

Returns a handle to the audio device in case of success, or otherwise (-1).

Notes: Not yet implemented!