# Live Video Encoder – Plugin API

**NOTE: SDK / Developer license is needed for this functionality.**

## Browser based usage with HTML and Javascript

The Live Video Encoder core components are available as plugins for several browsers:
Internet Explorer (IE), based on ActiveX, and NP-Plugin based browsers (Firefox, Safari, Chrome).
The same API is valid for integration into other languages, C++, .NET, Delphi, etc.
See the sample web page for functionality or contact us for further help.

## How to embed the encoder plugin into HTML

### Active-X Control

The Active-X Control is compatible with Microsoft Internet Explorer only. It is loaded by specifying the classid for the plugin. The plugin then may be accessed by Javascript via the ID specified, here "nanovid1".

```
Simple Version without auto-install:

<object ID="nanovid1" classid="clsid:E6F9EDB5-797B-4bc3-95E2-B7EA0886273B"
</object>

Advanced Version with auto-install from CAB file:

<object ID="nanovid1"
  classid="clsid:E6F9EDB5-797B-4bc3-95E2-B7EA0886273B"
  width="480" height="360"
  codebase="../packages/nanostream-1.5.0.6.cab#version=1,5,0,6"
</object>
```

Note: replace the `codebase` path and version with the cab file you received for installation.

### NPAPI Plugin

The NPAPI plugin is compatible with Mozilla Firefox, Apple Safari and Google Chrome.

```
<embed id="nanovid1" type="application/x-nanocosmos-LiveVideoEncoder" width=480 height=360>
  <noembed>
    Can't embed plugin.
  </noembed>
</embed>
```

It may be used with an <object> tag as well.

### Browser independent plugin loading

The object tags may be nested to support all browser types:

```
<object ID="nanovid1"
 classid="clsid:E6F9EDB5-797B-4bc3-95E2-B7EA0886273B" width="480" height="360"
 <comment>
   <embed id="nanovid1" type="application/x-nanocosmos-LiveVideoEncoder" width=480 height=360>
     <noembed>Can't embed plugin.</noembed>
   </embed>
 </comment>
</object>
```

### License and other custom settings

The License key is a string which needs to be placed into the nanoLicense.js file.
The nanoLicense.js also contains the mime type and path to the Mozilla based plugin (XPI). The default path is "../packages/nanostream.xpi":

```
= Contents of nanoLicense.js =
// nanoStream License Code and custom settings
// replace the "nlic" string with your custom license
g_nanoLicense = "nlic:1.0:nanoLiveEncDemo:……… ";

// url to xpi installer package
var nano_xpi_source = "../packages/nanostream.xpi";
```

### Dynamic DOM Loading of the Plugin

The plugin may be loaded into the HTML Document (DOM) by dynamically creating it with Javascript:

```
function createNanoVideoObject()
{
    var nanovid = document.createElement("object");
    if(isIE)
     nanovid.setAttribute("classid","clsid:E6F9EDB5-797B-4bc3-95E2-B7EA0886273B");
    else
     nanovid.setAttribute("type","application/x-nanocosmos-LiveVideoEncoder");
    return nanovid;
}
```

## Accessing the plugin

nanoStream is then available as a DOM object with the id "nanovid1". The member functions and variables can be accessed as other Javascript objects, e.g. nanovid1.StartPreview().

## Plugin Interface

The nanoStream Plugin may be controlled from almost any language which is able to load the plugin.
The plugin exposes methods and properties to set options, start preview and encoding.

**List of Properties/Functions with example settings:**

```
// destination URL (Flash or Wowza Media Server, or local file)
nanovid1.DestinationURL = “rtmp://localhost/live+myStream”;
nanovid1.DestinationURL = “c:\temp\h264.mp4”;
nanovid1.VideoSource = 0;            // Video Capture Device ID
nanovid1.AudioSource = 0;            // Audio Capture Device ID
nanovid1.VideoBitrate = 400000;      // Video Bitrate (bits/s)
nanovid1.AudioBitrate = 32000;       // Audio Bitrate (bits/s)
nanovid1.VideoWidth = 352;           // Video Pixel Width
nanovid1.VideoHeight = 288;          // Video Pixel Height
nanovid1.VideoFrameRate = 15;        // Video FPS

// start the preview from the current camera
nanovid1.StartPreview();

// Start Encoding/Broadcast
nanovid1.StartBroadcast();
```

## Javascript Helper Functions

For simple usage, a set of Javascript functions is provided (nanoEncoder.js).

```
// setup a live stream to a Flash Media/Wowza server
SetEncodingUrl(“rtmp://myServer.com/live/myStream”);
SetVideoBitrate(400000);      // sets the bitrate to 400 kbps
OnStartPreview();             // calls nanovid1.StartPreview()
OnStartBroadcast();           // calls nanovid1.StartBroadcast()

Additional Configuration can be set with

nanoSetConfig();

Example: nanoSetConfig("H264Profile","Baseline");
```

## Javascript API (nanoEncoder.js)

```
// SetEncodingUrl()
// call this to set custom encoding URL here
function SetEncodingUrl(url)

// get current internal encoding url
function GetEncodingUrl()

// SetVideoBitrate()
// sets encoder bitrate (bits/s)
function SetVideoBitrate(bitrate)

function GetVideoBitrate()

// SetVideoResolution()
// sets video capture resolution (x/y)
// warning: resolutions are hardware/driver dependent!
function SetVideoResolution(x,y)

// SetVideoFrameRate()
// sets video capture frames/sec (e.g. 15, 25)
function SetVideoFrameRate(fps)

// Start/Stop Video Capture Preview
function OnStartPreview(doCheckFormat)
function OnStopPreview()

// OnStartBroadcast()
// starts encoding / network broadcast
function OnStartBroadcast()
function OnStopBroadcast()

// SetAudioSource()
// sets audio source device id (0...)
function SetAudioSource(audioSource) {

// SetVideoSource()
// sets video  source device id (0...)
function SetVideoSource(source) {

// event_OnStop
// called by plugin on stop events
function event_OnStop(e)

// set basic encoding options to init plugin
function nanoSetOptions()

// Set license string to unlock the API
function nanoSetLicense(license) // call this before Init
```

## New functions for Version 1.5

```
Special configuration data can be sent with the SetConfig method:
nanovid1.SetConfig(property,value);

For some functions wrapper functions are available.

// H264 Encoding Profile ("Baseline" or "Main")
function nanoSetH264Profile(p) {
    nanoSetConfig("H264Profile",p);
}

// H.264 I-Frame Distance
nanoSetH264IFrameDistance(50);       // Default=50

// special h264 settings
nanoSetH264Profile(0);               // 0 or “Baseline”, 1 or “Main”
nanoSetH264VlcMode(1);               // VLC mode: 0=auto, 1=cavlc, 2=cabac

// Constant/Variable Bit Rate
nanoSetConfig("RateControl",0);      // 0=auto, 1=cbr, 2=vbr

// set server authentication (wowza/flash media server)
function nanoSetServerAuth(user,pass) {
    nanoSetConfig("Auth",user+pass);
}

// SetVideoResolution()
// sets video capture resolution (x/y)
// warning: resolutions are hardware/driver dependent!
function SetVideoResolution(x,y) {
    nlog("SetVideoResolution "+x+"x"+y);
    nanovid1.VideoWidth = x;
    nanovid1.VideoHeight = y;
}

// SetVideoResize()
// sets video resize resolution (x/y) (output of encoder)
function SetVideoResize(x,y,enableResize) {
    if(enableResize) {
      nlog("SetVideoResize "+x+"x"+y);
      nanovid1.VideoResizeWidth = resizeWidth;
      nanovid1.VideoResizeHeight = resizeHeight;
      nanovid1.EnableResize = 1;
    }
}


// set deinterlacing mode
nanoSetConfig("DeinterlacingMode", m);             // 0=off, 1=auto, 2=always
nanoSetConfig("DeinterlacingMethod", m);           // 0=bob/weave, 1=blend, 2=filter, 3=edge
```