Faq
Frequently asked questions
howto, troubleshooting

Updated Nov 1, 2012 by theintencity

Flash-VideoIO FAQ

Is Flash application a good choice for VoIP?

Please see my blog article FAQ on using Flash player to make phone calls for answers to these questions:

1. Is Flash application a good choice for VoIP?
2. Will there be any performance degradation when the call goes through the following paths? (Flash Client -> Media Server ->RTMP to SIP Converter -> VOIP Server -> VoIP/PSTN Gateway -> PSTN Network -> Telephone)
3. Some experts says that the development in C or C++ is prefered for VOIP call to phone instead of Flash Player for performance reason. Is that true?
4. There are different media servers available. like Adobe Flash Media server (FMS), Wowza, Red5 etc. Which one is the best choice?
5. We are now in a confusion whether to develop our VOIP application in Flash technology or QT/Java/C#. What will be your choice?

Which version of VideoIO should I use?

You should always use the latest version in the downloads page. If you can compile from sources, and notice additional important SVN updates beyond the latest version, we suggest you compile from sources. There is a Makefile containing the compilation commands that you can modify and use on any platform using Flex SDK. We recommend using the latest SDK.

Does it support echo cancellation?

Yes. There are three SWF files present in the download archive - VideoIO.swf, VideoIO45.swf, VideoIO11.swf. These binaries are compiled for target player version 10+, 10.3+ and 11+, respectively. For a production ready web application you should dynamically detect the Flash Player version and load the corresponding SWF. An example is shown in test.html (search for DetectFlashVer?). Dynamically detecting and loading the right SWF allows you to use the best features for the given version of the end-user's Flash Player -- enable H.264/G.711 for 11+ and enable echo cancellation for 10.3+. If you statically include VideoIO.swf you will not be able to take advantage of the echo cancellation or advanced codecs.

Does it support RTMP? What about RTMFP?

Yes, it supports both RTMP (client-server media path) and RTMFP (peer-to-peer media path). Most of the tutorial deals with RTMP. If you set the `src` property to appropriate "rtmfp" URL and set `farID` to appropriate value before setting `play`, then it enables RTMFP mode for peer-to-peer media. Please visit How to do peer-to-peer video call? for details. Also some example applications such as iChatNow, Random-Face and Public-Chat listed on the project page use RTMFP using Adobe Stratus service.

Does it support web-to-phone calls and vice-versa?

Yes, partially. It supports Flash-to-SIP calls and vice-versa using an external SIP-RTMP gateway such siprtmp.py. You can use the extensive Javascript API to build your own web phone user interface including call control and a dial-pad for sending in-call DTMF digits. You *need* to use your own PSTN termination provider that can terminate your SIP calls and translate to PSTN, and vice-versa. Please see How to do SIP-based VoIP calls? for details. In summary, your termination provider must support Speex voice codec, and understand SIP, SDP and RTP.

Can I use it to build a video call application on Facebook using FBML?

Yes. Please see the Face Talk application. Facebook's FBML mangles Javascript. Due to this the API used to interact between Flash Player and Javascript breaks. A friend of mine and I worked on a making Flash VideoIO compatible with Facebook.

How do I embed it in another Flex application?

Starting with version 1.5, I have added support in VideoIO so that you can embed it in your another Flex application using SWFLoader. The VideoIO.swf detects that if it is the top level application then it enables ExternalInterface? based communication with Javascript, otherwise it allows your top-level Flex/ActionScript? application code to communicate with it directly using similar method and event names as the Javascript API. Until I formally document the API, you can see the example MXML application code for Flex Builder 3/SDK 3 or Flash Builder 4/SDK 4. Since VideoIO is compiled with Flex Builder 3/SDK 3, it works very well if your application also uses SDK version 3. There are some compatibility issues if your application uses Flash Builder 4/SDK 4, e.g., with respect to the sizes of control bar and buttons, and event dispatched by VideoIO to application.

How do I implement text chat?

All our existing applications use external signaling channel such as websocket or channel API to implement any text chat messaging. Please see the example source code in SVN for details.

If you wish to do the text chat within VideoIO without external mechanism: In the client-server case you can use call method and onCallback callback to implement a server mediated text chat within RTMP. In the group communication case you can use the post method and onPostingNotify callback to implement the group text message. In version 2.6 we added a new API, sendData method and onReceiveData callback that allows sending some text data in a publishing stream which is received by all playing stream.

To test this feature, use a local RTMP server, open two windows with test.html page, set "src" property to `rtmp://localhost/myapp?publish=live` in first and `rtmp://localhost/myapp?play=live` in second, to enable streaming from first to second. Then in the first browser, in the call input box of bottom right corner, type `sendData('this is some text')` and press enter. In the second browser, you should see the onReceivedData callback with this text in the text area box. Note that the sendData can only be called from publishing VideoIO and onReceivedData is only invoked on playing VideoIO side.

This feature can be used for both client-server and P2P RTMFP communication to implement text chat as follows. Suppose in a two-party call, if user A has video1 and video2 for publish and play respectively, and user B has video3 and video4 for publish and play, then user A can send message by calling getFlashMovie("video1").sendData("some text message") and on user B's side onReceiveData(event) callback is invoked which has event.data as "some text message" and event.objectID as "video4". When user B wants to send a text message it uses getFlashMovie("video3").sendData("another message") and on user A's side onReceiveData(event) callback will have event.objectID as "video2".

The API description is as follows until it appears on the tutorial pages. The "sendData" method can be used to send some text in a publishing stream to all the other subscribed playing streams. This is particularly useful for P2P RTMFP mode using Stratus service where you cannot use the call() method and implement a server side feature, but you need an end-to-end text/data transfer mechanism.Unlike the post() method that works in a group, the sendData() method is useful for client-server and peer-to-peer streams. The other end receives a onReceiveData(data) callback method invoked when it receives the sendData command in the stream. For a one-to-many stream, the publishers calls sendData and all the players receive onReceiveData callback. You can call sendData only if VideoIO is publishing, and will get onReceiveData only if VideoIO is playing. It is up to the application to define the meaning of the text data send and received via this method.

How do I implement picture-in-picture mode?

The picture-in-picture mode is not planned to be explicitly implemented in VideoIO.swf because we believe you can achieve the same effect by embedding VideoIO in another Flash application or in HTML.

We created an pip.html and VideoPIP.html to test out this feature. You need to use Flash Player 10.3 or later, and use rtmplite's rtmp.py server for a quick test. You can view the source and incorporate similar code in your HTML or Flex application for picture-in-picture mode.

For embedding in HTML as shown in pip.html there are certain restrictions e.g., the minimum dimension of VideoIO must be 215x138 and full-screen mode is for each VideoIO. To avoid these restrictions you can embed the two VideoIO.swf in another Flash application as shown in VideoPIP.html.

How do I use H.264 encoder?

Flash Player 11 (beta) comes with H.264/Avc encoder, whereas the decoder is available in Flash Player 9 or later. Flash-VideoIO 2.4 or later supports H.264 encoding. Please make sure to use VideoIO11.swf that is compiled for target player version 11. If you use the test.html page, it automatically detects your Flash Player version and uses the correct VideoIO. You can right click on the video box to see the Flash Player version. On Chrome browser which comes with a built-in Flash Player, even if you install player version 11, it will prefer to use the built-in version. To change this open "about:plugins" from your Chrome address bar, click on "Details" on top right, and disable the built-in Flash Player that came with Chrome, so that your other installed Flash Player is used. Once on the test.html page, change the "videoCodec" property to `H264Avc` and press enter. The value must be exactly `H264Avc` and not alternatives such as `H264/Avc` or `H.264Avc`. Then start a locally running RTMP server, and set the "src" property to `rtmp://localhost/call/123?publish=live` and press enter to start publishing in H.264. Open another tab in your browser and visit the same test page. Set the "src" property to `rtmp://localhost/call/123?play=live` and press enter to start playing the stream published by the previous tab. You can see noticeable difference in the quality of the video played. In particular, the blockiness of the video disappears when using H.264. If you enable "cameraLoopback" property on the publishing side, you will still see blockiness of the video; perhaps Flash Player still displays the local loopback video using Sorenson, but sends using H.264 on network when the video codec is set to H.264. Try setting "cameraQuality" property to 80 and press enter for a reasonably good quality value.

How do I use G.711 (pcmu/pcma) audio codec?

Flash Player 11 (beta) comes with G.711 encoder and decoder. Flash-VideoIO 2.4 or later supports this. Please make sure to use VideoIO11.swf that is compiled for target player version 11. By default the "codec" property is set to `Speex`. You can change this to `pcmu` or `pcma` and press enter to change the encoder when publishing. The receive side automatically decodes any received audio codec.

How does it zoom video to fit in given display size?

The current implementation has an auto-zoom behavior as described below. I will add a new "zoom" property to control the behavior in future.

The behavior is that if the available video (either from camera or remote publishing stream) is in aspect ration 4:3 but the display ratio is not the same, it displays the central part of the video by zooming in as needed.

Periodically the publishing side sends a NetStream?.send("setVideoSize", width, height) based on the display size from publishing side VideoIO. This is received by the playing side to adjust the publisher video size instead of using the publisher's camera size as the video size. This is used to chop off the displayed video on playing side as well. This works well in a video conferencing scenario where the player can see less than or equal to the publisher. And allows the video conferencing endpoints to change VideoIO size to anything.

Unfortunately this results in a very complex behavior in case of non 4:3 aspect ratio especially with HD ratio in full screen. For example, suppose camera capture is 320x240 (ratio 4:3) but publisher has monitor resolution of 1280x720 (HD ratio). As per the auto-zoom logic, this zooms the central 320x180 in full screen mode on publisher side. However the full 320x240 video is sent in stream. On the receiver side suppose the monitor resolution is 720x480 (ratio 3:2) then the full incoming 320x240 stream is displayed as distorted (vertically shorted from 4:3 to 3:2) in full screen before setVideoSize is received. After setVideoSize(1280, 720) is sent by publisher and received by player, the incoming 320x240 stream treated as central 320x180 stream which is further zoomed out so that central 270x180 is expanded to fill full screen. As you can see, this could result in significant loss of video on the sides from 320x240 to central 270x180).

To avoid the loss of video on the sides, especially things such as TV channel icons and sub-titles, I will add an "zoom" property which can be used to configure the zoom behavior, e.g., null -- don't do any zoom but allow distortion if aspect ratio does not match, "in" -- perform zoom as per the mechanism mentioned above, "out" -- instead of zooming in, do zoom out, so that the full video is displayed even though sides may be empty (black or white) if the aspect ratios are different. Both "in" and "out" values are used to prevent any distortion in video, if video and display sizes are different.

How do I append instead of overwrite when record is true?

We have added a new property "recordMode" based on a patch by Qi Wenmin in SVN r49. This will become available in version 3.3 and later, until then feel free to use the SVN copy. The property can be used to set the record mode to append. Please see the API description of recordMode for details.

It works on my local computer but not on the Internet. Why?

VideoIO itself is a tool, not a service, and you need a service to make it work completely. It can be used to connect via RTMP (TCP) or RTMFP (UDP). Both these require a service (or servers) that enable the connection between two VideoIO instances. Hosting a server costs money. Luckily, Adobe provides a free Stratus/Cirrus service for a subset of RTMFP, which we use in our free demonstration applications (vvowproject, public-chat, AIRphone, etc). Unfortunately, RTMFP does not work across all kinds of network devices. This forum post provides more details on why this happens and what you can do about it. In summary, this is not a VideoIO issue but a Flash Player issue, and can largely be solved in your application that uses VideoIO and with your hosted service. You start with RTMFP, detect that it works or not, and fall back to RTMP if needed

Comment by evgeniiml, Mar 9, 2012 Delete comment

I have problem with "sendData" method it does not work for me in your test example. It fails with: "invalid method call ..." I also try to do it as in docs with 2.6 and 2.7 version, but recive nothing on playing side.

Comment by elektrosex, Mar 28, 2013 Delete comment

Please try sendData("this is some text") instead. Should work

Comment by mohsen.mhndm, Aug 21, 2013 Delete comment

How I can record .mp4 files ? or .ogg I don’t like ,flv files

Enter a comment: Hint: You can use Wiki Syntax.

Submit