STEP BY STEP SMART-SRT MANUAL FOR PC

Prepare everything you need

- Download the **SmartSRT Control** installer from https://www.todostreaming.eu/es/download.html and install it (Windows 7 or higher). You will need NET Framework 4.6.1 that will be automatically installed in your system if you don't have it yet.
- Download the **SmaRT Live Encoder** that you will use to send SRT (also for Windows 7 or higher, 32 and 64 bits). If you dont know what your system is, download the 32 bits version by default.
- Download the **SmaRT Receiver**, that you will use to receive SRT, and send TS-UDP inside the same machine to player softwares such as VLC, OBStudio or vMix.

We register on the service SmartSRT

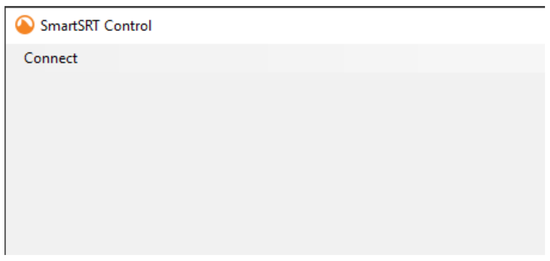
You can ask for a free DEMO to sales@todostreaming.es
We will give you the folowing access data:
- Server Num ID.- An integer number that says which SmartSRT server will be used. (ie: 1)
- E-mail.- Used in the Log In process to your account. (ie: user@gmail.com)
- Password.- Used to enter your account. (ie: 12345)

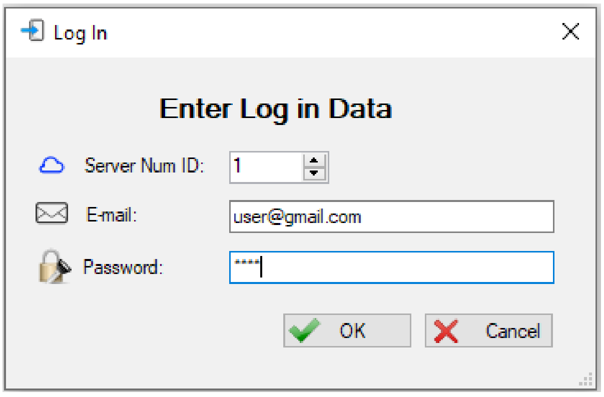
Configure the sender side

With everything necessary installed, connected to the network and plugged into the electricity, we will start with the configuration on the **sender** side.

1) Launch the **SmartSRT Control** app.

A completely empty window will appear.

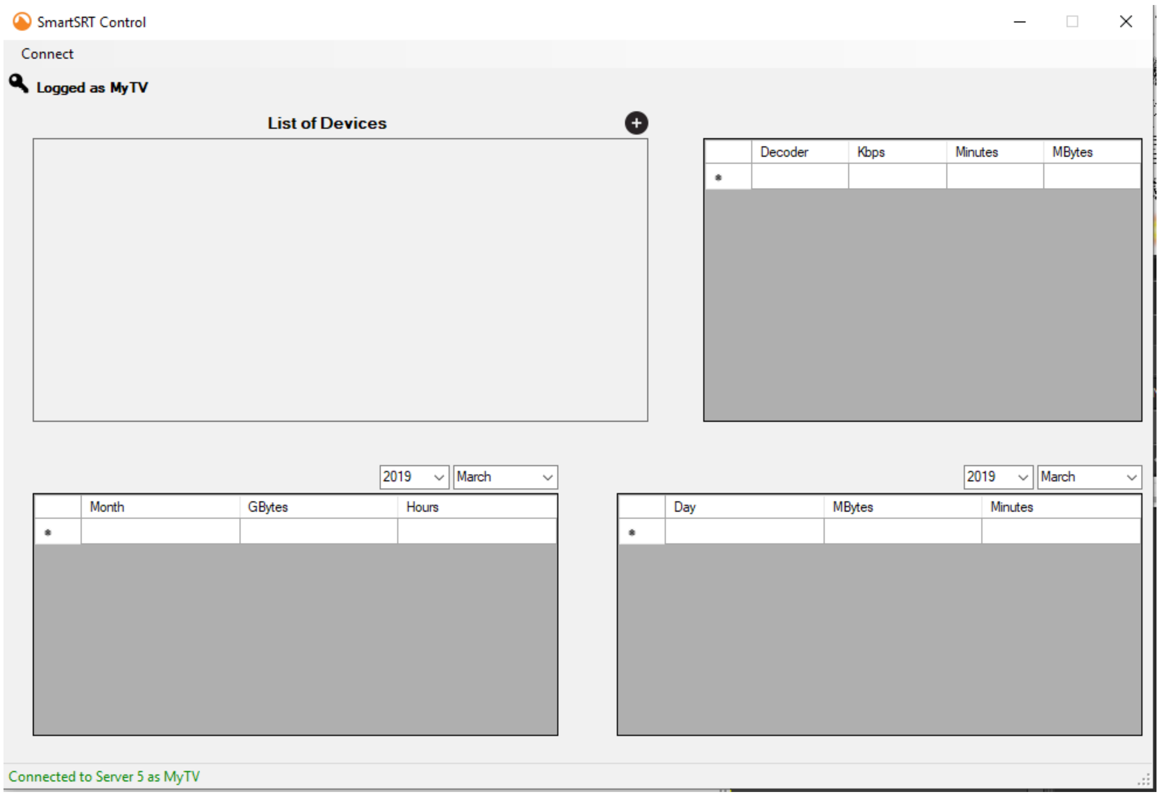




Server Num ID: 1
E-mail: user@gmail.com
Password: ****

Click on "Connect" menu, and then "Server SmartSRT". Fill in the log in data.

The window will be filled by several blocks:

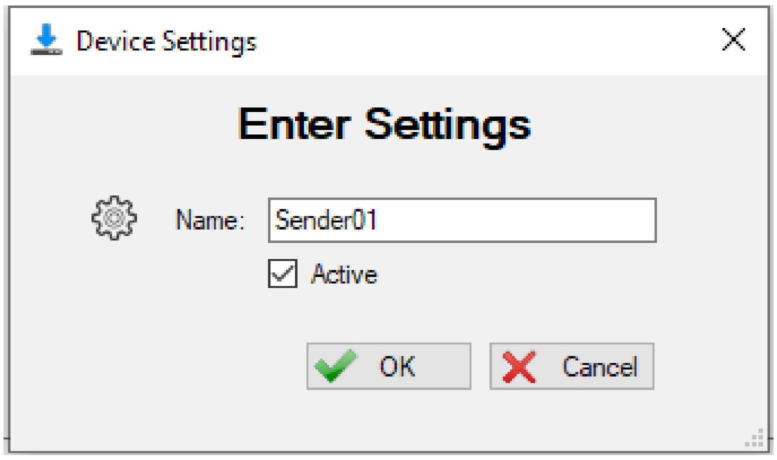


Connected to Server 5 as MyTV

If we log in correctly, and have a direct connection to the Server, a **green** message will appear in the lower status bar indicating it. Otherwise a message will appear in **red** with the error.

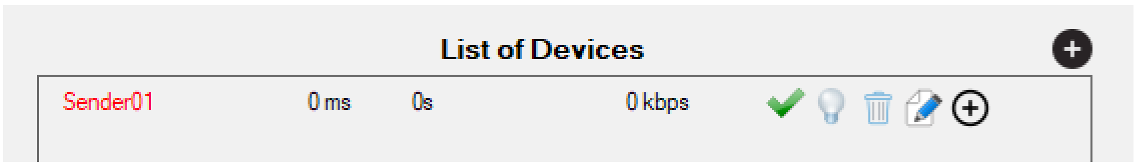
2) Register the **sender** device

In the block with the heading "Device List", in the upper right part, there is an icon in the form of a black circle and a white cross inside. If we place the mouse cursor over it, a label appears with its "Add a New Sender" function. Click on it, and a popup window appears to fill in the sender data.



Name: Sender01
☑ Active

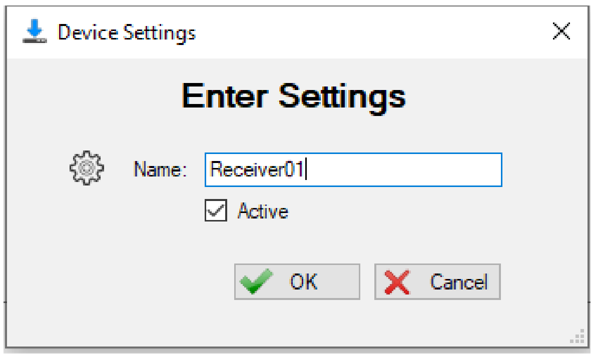
Put the easier name to remind. You can change it at any moment. The new registered device will appear in the Device List.


List of Devices

Sender01	0 ms	0s	0 kbps	✓

If we place the mouse cursor over the different elements, a message will appear that identifies what it is, and if the cursor changes shape to a hand, it will indicate that this element is an active button, which allows to change things on that device.

We will describe them from left to right, passing the mouse cursor over them:
- Copy encoder smart url: It appears when you pass over the name of the device. It's an active button, if we click on it, does not seem to do anything. It will copy to the clipboard the **smart url** address that we will copy in the **SmaRT Live Encoder** that will act as a **sender**.
- Buffer delay: It indicates the maximum milliseconds of delay in this connection.
- Time connected: It says the uninterrupted time the device is connected to the SmartSRT server. In days, hours, minutes and seconds (ie: 3d:21h:34m:15s).
- Bitrate in Kbps: Shows the bitrate calculated by the local multiplexer. It is an orientative and inaccurate measure, which with the minutes adjusts to its real value.
- Activate/Deactivate device: This element is an active button, when it is a **green** check icon, allows us to deactivate the device, and in the form of a **red** cross, it allows us to re-activate it. Before changing, a pop-up window will ask if we are sure to do it. A disabled device can not connect to the SmartSRT server while is in that state, this can be used to control the access of each device to the distribution network in real time. If we deactivate a sender, all receivers connected to it will also be deactivated.
- Connected/disconnected device: A bulb that turns on or off, this indicator is very clear about the status of the sending or receiving equipment at all times.
- Delete device: It is an active button that may erase a device in the network. The erasure is definitive, so a window will appear to confirm our delete intention. If we delete a sender that has receivers, all the receivers assigned to it will also be permanently deleted. Handle this button with care. It is preferable to use the disable button explained above, to regretting and reconfiguring the deleted computers from scratch, since smart URLs are generated randomly, and never are the same.
- Edit device: It is an active button that shows a window where you can edit the status of the equipment and its name. You can change these things at any time whenever you want.
- Add a New Receiver: It is an active button in the form of a circle with an internal black cross, which only appears to the right of the senders, and allows us to register receivers that will connect to this sender. The window and registration process is the same as that of the receiver. Let's click on it, and we will add 2 receivers to this emitter.

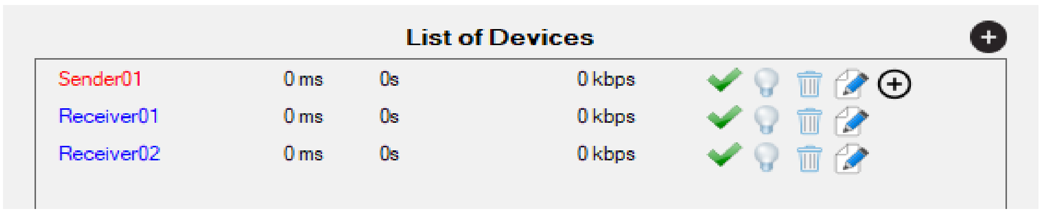

Device Settings

Enter Settings

Name: Receiver01

☑ Active

At the end of the process you will have something like this.



List of Devices

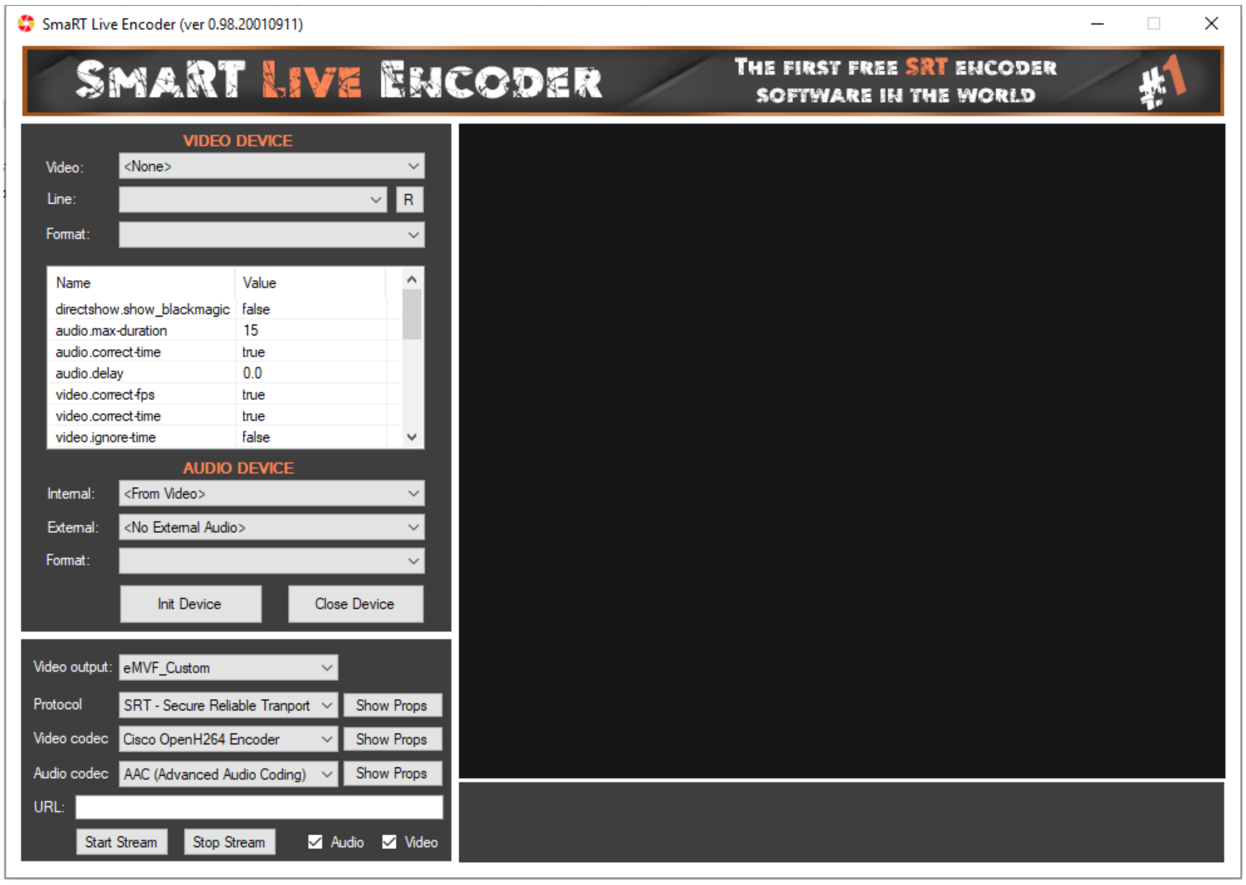
Sender01	0 ms	0s	0 kbps	✓
Receiver01	0 ms	0s	0 kbps	✓
Receiver02	0 ms	0s	0 kbps	✓

You can add senders and receivers, according to your needs and your distribution network. The sender will appear first and below all its associated receivers. At a glance we can control all of them in real time.

You can add, change and remove network elements in production at your own will.

3) Configure of **SmaRT Live Encoder**

When running the executable file (smartencoder.exe), after a few seconds, it will show a window like this one.



SmaRT Live Encoder (ver 0.98.20010911)

SMART LIVE ENCODER — THE FIRST FREE SRT ENCODER SOFTWARE IN THE WORLD #1

VIDEO DEVICE

Video: <None>

Line:

Format:

Name	Value
directshow.show_blackmagic	false
audio.max-duration	15
audio.correct-time	true
audio.delay	0.0
video.correct-fps	true
video.correct-time	true
video.ignore-time	false

AUDIO DEVICE

Internal: <From Video>

External: <No External Audio>

Format:

Init Device | Close Device

Video output: eMVF_Custom

Protocol: SRT - Secure Reliable Tranport

Video codec: Cisco OpenH264 Encoder

Audio codec: AAC (Advanced Audio Coding)

URL:

Start Stream | Stop Stream | [✓] Audio | [✓] Video

The configuration is done in the left area, and is done line by line from top to bottom, since what we select in the upper options affects the possible options displayed in the lower options. Therefore, we start with the video block, in the Video option, where we can choose the video source that we are going to capture. Among the options, you will see video capture cards (Black Magic, AJA, Bluefish, Dektec, Deltacast, Magewell, Streamlabs), DirectShow devices (webcams, generic capture cards), virtuales devices (vMix Video, Medialooks Webcapture, Medialooks Screencapture) and NDI sources.

Depending on the selected video source, it will appear in Line, the possible possible inputs. This will be detected immediately, or in some cases we will have to press in R to reload the new entries (such as NDI sources).

Then we will see the different formats (Format) that are accessible on that input. Some source devices, such as the new Black Magic cards, can be left to auto, so it autodetects the format of the incoming source. Below there is a table of advanced settings, which is only convenient to adjust, if you know what you are doing. Otherwise leave to defaults.

The audio zone is self-explanatory. By default, the audio is taken from the same video source (Internal = <From Video>). External audio can be added to the video, from additional sound cards.

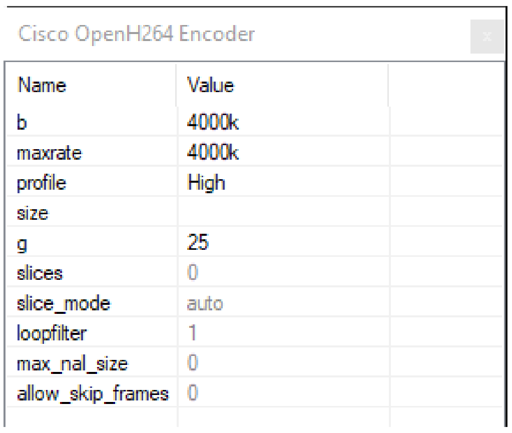
NOTE: If you are going to use Transport Stream Over UDP (TS-UDP) in vMix to get the stream, it is important that the audio format chosen is 48000 Hz, 2 Ch, 16-bit. Otherwise it is better to use the VLC plugin that supports other different formats.

Once we have configured the capture device for video and audio, press Init Device, to start it, and the corresponding video capture will appear on the right.

In the lower block, there are the settings of the streaming output that we are going to configure. In the Video Output option, by default it will appear in eMVF_Custom, which leaves the original screen size of the capture, but we can change that size to our convenience.

SmaRT Live Encoder is able to broadcast in various streaming protocols (RTMP, UDP, UDP / DVB compliant, SRT, SRT / DVB compliant, RTP / ProMPEG, RTSP). By default, the SRT / DVB protocol is selected, which is what we use in the SmartSRT service. On the right, on the Show Props button, a table of advanced self-defined options is displayed that should only be changed if you know what you are doing. You can leave on default values.

The default video codec we are going to use is the Cisco OpenH264, which uses less CPU and offers an acceptable quality. We can adjust it in Show Props.


Cisco OpenH264 Encoder

Name	Value
b	4000k
maxrate	4000k
profile	High
size	
g	25
slices	0
slice_mode	auto
loopfilter	1
max_nal_size	0
allow_skip_frames	0

This is an example of HDTV broadcast where is convenient to use a bitrate (b) of 4000 Kbps, constant so let's fix maxrate. We will use a High profile for H264, and a GOP (g) of 25 frames. For SDTV we recommend 2000 Kbps.

NOTE: SmaRT Live Encoder, is a commercial freeware, so it can not carry GPL codecs. However, the user, under his own responsibility and risk, can change in DLLs the libcodec-58.dll library file to support the GPL x264 and x265 codecs compiled by himself, but he will not be able to distribute it to third parties.

NOTE: If you are going to use Transport Stream Over UDP (TS-UDP) in vMix to get the stream, the supported codecs are H264 for video, and AAC for audio. If you want to use different codecs, it is convenient to use the VLC plugin.

Finally we have the URL field, where we will paste the **smart url** address that we can copy to the clipboard from the device panel of the **SmartSRT Control**.

Click on Start Stream button, and the SRT broadcast will start.

NOTE: The rest of the protocols use the same format as ffmpeg for those same protocols. You can freely use this software to send streaming in these protocols to any server that supports them. TodoStreaming does not support the use of servers others than those of SmarSRT.

At the start of the streaming, the lower right area goes orange, and it will show the capture logging. If everything has been done correctly, in the SmartSRT Control panel you will see how the sender starts counting seconds on its connection as well as the rest of the data.

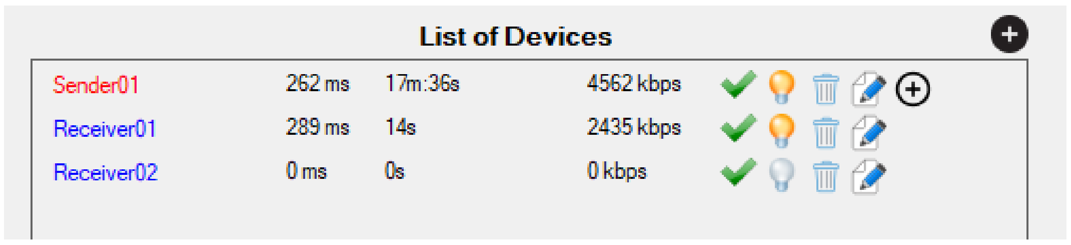
4) Configure **SmaRT Receiver**

We are gonna receive the stream sent by SmaRT Live Encoder, or from any other SRT source, such as SmartSRT Cam for Android or iPhone (our mobile apps broadcasting live video) or from a RaspberryPi upload (Step by Step manual for RPi).

SmaRT Receiver is a mini-app that we can leave running in the desktop indefinitely. Even if the signal is dropped, this mini-app will reconnect it immediately when it returns, indicating it in the upper zone.

By default comes ready to get the signar we set on Streaming URL, copied and pasted form the SmartSRT Control, and send it locally to the same PC on TS-UDP (default port 1234).

Once configuration is finished, click on **Listen** button. After what we will see, in the **SmartSRT Control Device List**, how the numbers of the connected device change.


List of Devices

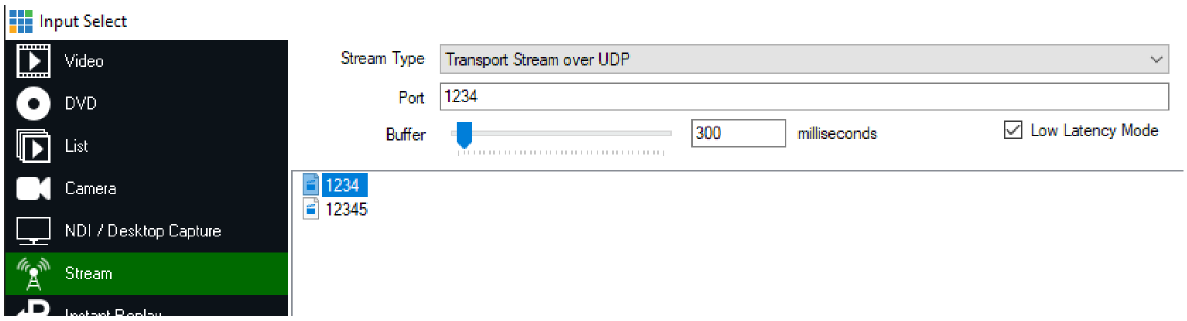
Sender01	262 ms	17m:36s	4562 kbps	✓ ●
Receiver01	289 ms	14s	2435 kbps	✓ ●
Receiver02	0 ms	0s	0 kbps	✓ ○

In the same way, all the receivers we need in our distribution network can be programmed and connected. You can also run several sessions of this mini-app on the same machine to retrieve several SRT signals and forward them to different UDP ports on that PC.

To get the UDP local stream in the same PC, we can use VLC opening a network stream on udp://@:1234 (if the TS-UDP used port is 1234). To low the latency of the player just change the network-caching to 300 ms (inside VLC).

OBStudio also has a VLC Video source, that allows connecting to the UDP stream the same way, and also lowering the network caching to 300 ms.

vMix, apart from the VLC plugin, owns a native input for Transport Stream over UDP, where we only have to set the port number used.



Stream Type: Transport Stream over UDP
Port: 1234
Buffer: 300 milliseconds
☑ Low Latency Mode

1234
12345

Using buffers less than 300 ms, can make playback unstable, in all players.

Latency, buffers and stability

It is important to know how parameters determine the total latency of each receiver with respect to the original signal, and how they also influence stability

The SRT is the protocol used by all the devices that connect to the SmartSRT server through the public Internet. This protocol, although it is based on UDP, has an advanced mechanism to recover lost packets, giving it a reliability similar to TCP.

- The Sender Buffer delay (appears in theSmart Controla app, List of Devices) (ie: 262ms)
- The Receiver Buffer delay (appears in the same site) (ie: 289ms)
- Playback Buffer (seen above) (ie: 300ms)
- Converting protocols + multiplexer at SmartSRT server = 200 ms.
- Latency of encoder (when compressing). SmaRT Live Encoder has 300 ms.

262 ms + 289 ms + 300 ms + 200 ms + 300 ms = 1351 ms (1.35 seconds max)

So in our example the max latency on **Receiver01** will be around 1.35 segundos. However in real conditions when we did the example, the real delay was 1.10 seconds.

SRT is a low latency protocol, where the values of the buffer delay range from 50 ms to 5000 ms, depending on the PING and the packet loss. But although the default system calculates these buffers, there is a way to force them to a specific value, if for example we want to lower it, because we do not care that artifacts appearing in the video from time to time, and we are interested in having a lower latency, or if on the contrary we want to increase its value, because we are much more interested in the stability of the long-term issue.

To force these values, it is necessary to change the whole number that appears at the end of the Smart type URL, which by default is a 0 (automatic buffer).

In our example it was: **smart1://IoYkkChsMKCyLkJH/0**
So if we want to use a value of 50 ms, our URL would change to:
smart1://IoYkkChsMKCyLkJH/50

If we set a value outside the allowed range, between 50 and 5000, the system will set the limit value.

Our implementation of SRT, ends the communication between 2 parties, only when there is no transmission of any package for at least 3 seconds in a row. And when running on UDP, it only depends on the buffers and network conditions to recover lost packets. In case of not being able to do it, it will not drop the connection, but you will simply see artifacts in the reception.

In case of experiencing unwanted artifacts in reception, we recommend raising the buffer to twice the value created automatically. Using a buffer twice the automatic, allows us to be immune to data loss rates of up to 10%, conditions that would make a game over on the rest of streaming protocols.

Next we are going to expose an example PING table, together with buffer values and the final maximum latency, as reference values.

PING (ms)	Sender Buffer	Receiver Buffer	Player Buffer	Max Latency
33	100	100	300	1.0 s
166	500	500	500	2.0 s
230	750	750	500	2.5 s
500	1500	1500	500	4.0 s

As a final observation, I would add that buffers within the same network are not related to each other, so they do not need to be the same, since each device will be in different network conditions, and at a different distance from the SmartSRT server. If all the receivers dependent on the same sender experience artifacts, it is very possible that the problem comes from the connection of the sender to the server, so the buffer of the sender is the one that we would have to be raised to avoid this inconvenience.

TodoStreaming March de 2019