HOW TO USE ONPREMISE SRT SERVER

Introduction

Before we start explaining how to use *OnPremise SRT Server*, let's make an introduction to make clear what it is, and what it has been created for.

The motivation to create *OnPremise SRT Server,* was to ease the task of building a terrestrial deployment of repeaters for Radio and/or Television, with feedbacks and real-time control for each of its elements, using the public Internet network, through the secure and resilient **SRT** protocol.

Why SRT?
- SRT is an open source protocol, very widespread today, and present in most software and hardware on the market (VMix, OBS, VLC, Wirecast, Larix, Kiloview, Haivision, Teradek, Truen...)
- SRT can be set up for both low latency and high resilience to network congestion and packet loss. When RTMP or HLS cannot connect at all, SRT can still do so, with a buffer delay of less than 8 seconds. It is suited for very large RTT values (ping above 500 ms), or when packet loss is greater than 10%.
- SRT provides statistics of the medium through which it is transported, allowing you to adjust its parameters to optimize to the use case.
- SRT naturally carries MPEG-TS, being totally codec agnostic. It can therefore transport MPEG-2, H.264, H.265/HEVC, H.266/VVC, etc. It can also transport SPTS or MPTS. SCTE data and other signaling, and anything that can travel within a Transport Stream.
- SRT allows point-to-point security, using 128, 192 and 256 bit AES encryption.

For this task, *OnPremise SRT Server has been* equipped with an architecture where IP inputs and outputs can be easily connected to each other. Thus, there can be several inputs, and each of them can copy its content to one or more outputs at the same time, without any delay.

In addition to the logical SRT inputs and outputs, which are the basis of the server, it has been provided with pull inputs from other protocols, with the idea of easing the contribution of channels from other servers operating in those protocols, such as UDP (unicast or multicast), RTMP, RTSP, HLS, DASH and HTTP. This does not mean it is an RTMP server, but merely a gateway that can pick up signals in these protocols from other servers. It has also been used to add, apart from the usual SRT outputs, UDP outputs (unicast and multicast) useful in some IP headers, and RTMP to be able to send copies to Youtube, Facebook and other servers that still use this protocol for signal contribution.

OnPremise SRT Server repeats exactly the incoming stream at each output with no modification at all, with the same PIDs, SDT and EIT tables, EPGs, etc. It does not recompress audio or video, nor does change on them, because that is not its purpose, so that, what is sent to it, is what it will reach to all of its outputs with no modification at all.

The *OnPremise SRT Server* panel shows at a glance, all the components of the contribution and distribution network (senders, receivers, etc.), their real-time second-by-second status and valuable statistical information. There is absolute control over each of them, you can access their remote panels, disable or enable them with a simple click, or delete them forever, and also edit their connections. In addition, all inputs can be monitored through a watchdog, to know if the audio is muted, which is useful to detect failures in the feedbacks, or in the sources, warning through a red visual alert. It can track both connected and disconnected components, unlike a traditional media server.

The fact that it has been designed to be deployed in situ (in the studio or headend), and the type of resources required for its operation, means that it only runs on real bare metal computers. Although the ISO image can be installed in a virtual machine, it will never run in this one, and it is something that we do not plan to do at any time in the future. We recommend its installation on computers with at least 4 CPU threads at 1.5 GHz or more, on 64-bit architecture and 4 GB of RAM minimum. Read the HARDWARE_SPA.pdf document about the recommended hardware for installation.

Each element participating in the network via SRT establishes a point-to-point connection to the server, which is unique, fully secured and monitored separately from all others. Therefore, this connection is done, through a UDP port that is totally different from any other connection. Therefore, we do not multiplex different SRT connections on the same UDP port for this reason, nor do we plan to do so in the future, because that is not the purpose which *OnPremise SRT Server* has been created for.

This software has a simple update option. When there is a new version, the software downloads it, but does not apply it until the user clicks on the update button. The update can be performed without stopping the server operation. The system hot reloads, and all the devices reconnect to it in less than 2 seconds. Updates will correct bugs and add new functionalities of interest for the purpose of this software.

OnPremise SRT Server exposes a REST API, so that it can be integrated with other programs or external processes that want to work with it. This API is self-documented with Swagger 2.0, although it will be the subject of another document to explain how it works.

OnPremise SRT Server is software that is installed together with a 64-bit Debian Linux operating system, and is accessed via a web browser from any other computer connected to the network. In this way, stable and secure operation can be guaranteed for years in a 24/7 continuous working environment.

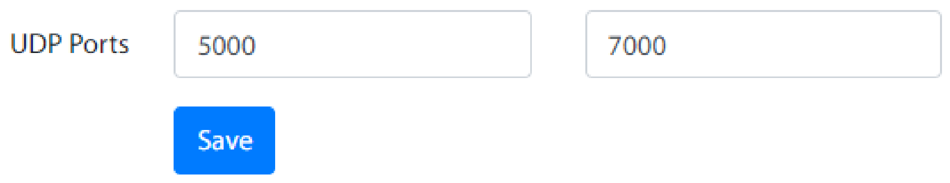
In this manual, we will explain how to create good SRT connections between the different components of the network and this server. We will work with the most typical components on the market, and we will face the most normal situations in a studio or in a headend.

Let's get started.

Initial system configuration

Before starting to create links, we have to set up the server configuration, once the software has been installed, according to the instructions in the INSTALL_SPA.pdf document.

The first thing to do, is to set a fixed local IP for the server. It is advisable to use one that is outside the DHCP pool, i.e. that is not one of those assigned dynamically by DHCP, and that is free. For example, if DHCP delivers IPs from 192.168.1.100 onwards, we can choose a free one below that value, such as 192.168.1.68 . The network administrator must know which parameters must be set up here. To do this, once logged in as admin in the server panel, go to the Network section, click on LAN/WLAN Interfaces and set the Mode to Static, the IP and mask to Address/Mask, in our example it would be 192.168.1.68/24 and its corresponding Gateway. It is also important to set at least one DNS in the LAN/WLAN DNS section.
Next on the same screen, go to Other Options, and limit the UDP ports that we are going to use on this server. It is more than enough to assign a range of 2000 ports, for example 5000-7000 or 7000-9000. For that, we set this range in UDP Ports and press Save.


UDP Ports 5000 7000

This way, we are sure not to create a SRT Listener connection outside this range, since the software will warn us.

To make those ports accessible from the outside of our private network, it is necessary to create a NAT rule in the router, that directs that range of UDP ports (not TCP) to the local IP of our server. If you don’t know how to do it, ask your network administrator to do it for you.

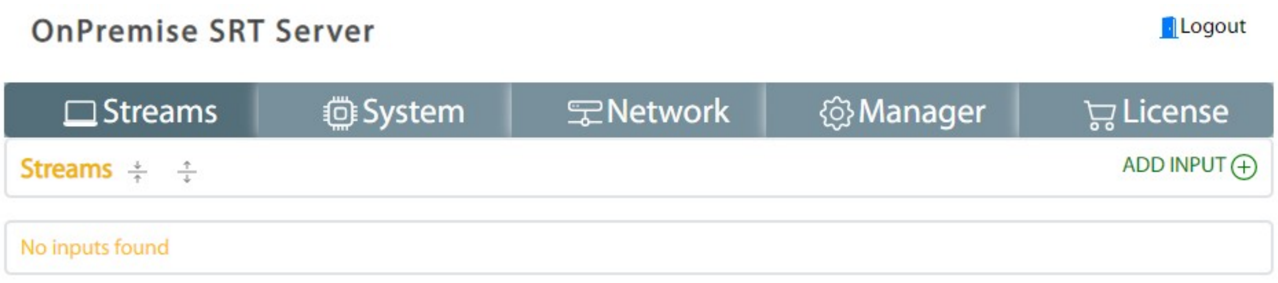
Once all the Network section is set up, go to the System section and click on the Reboot button, and then on Logout. The equipment will restart with the new network parameters, so in your browser you will have to enter the new fixed IP of your server, to be able to log into it.

The last step is to have a fixed public address for your Internet connection. If your public IP is not fixed, it is convenient to create an account at No-IP.com or Dyndns.com, set a hostname for your dynamic IP connection and set it up in the DDNS block. In this way, your address from the outside would no longer be a changing IP, but a concrete and fixed address such as: srtlinks.dyndns.org.

The system is now ready to create all the links we need, from inside and outside the local network. Let's get started.

SRT Links

From now on, all of our work will be done in the Streams section. At first, it will be completely empty and will look like this.



No inputs found

To start creating our network, we have to begin adding at least one INPUT. This is done by clicking on ADD INPUT, which will open a window that we will have to fill in with several data. But before that, we are going to explain some general aspects of this section.

All the IP video sources are INPUT, and can be from a video encoder or other server that sends a signal thru SRT, or a server with other protocols which we can pull the signal from. In any case, there are a series of parameters common to all the INPUTs:

Protocol: communications protocol used (SRT in our case).
Name: will be used to recognize the source of this signal, with a descriptive name that will be displayed on the panel. If left blank, it will take the value Channel 1, and so on.
Provider: will not be displayed in the panel. If left blank, it will take the value TodoStreaming.
Location: will not be shown in the panel and is optional, but it can be useful to know the geographic location of our INPUT source.
Remote: it is also an optional field, in it we can put the http URL to access the control panel of the source device. If we fill in this field, an icon will appear, where we can click to go directly to the control panel of this device.
Wtdg Timeout: It is a numeric value between 0 and 3600 seconds, which indicates the time in seconds of audio silence that will trigger the watchdog alarm of this INPUT.
Watchdog: is a checkbox that, if activated, triggers the alarm described above.

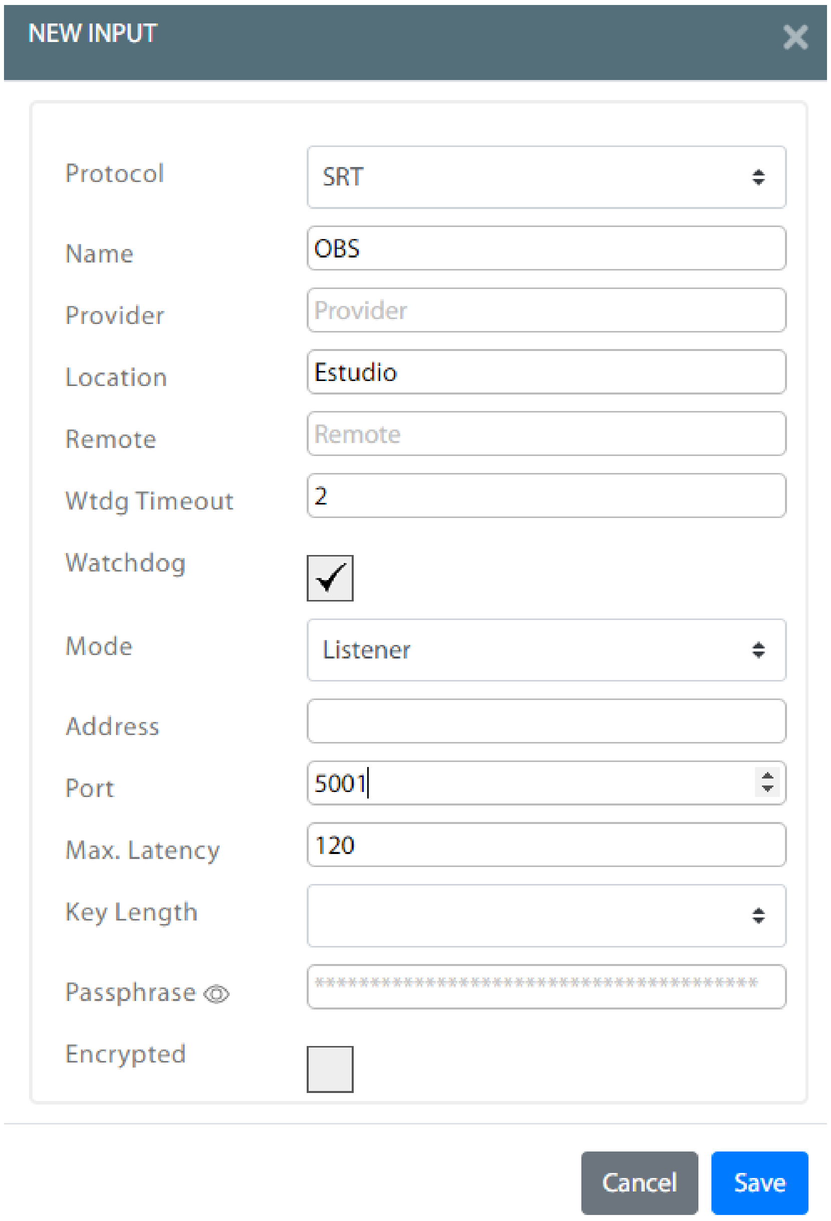
The rest of the fields of an INPUT are specific to each protocol. We will first explain the SRT protocol, which is the basis of this server.

SRT has 3 modes (caller, listener and rendezvous). In this manual we will only deal with the first two. Always, in a point to point SRT link, one end will be caller and the other one will be the listener. For simplicity, the INPUT server part will be a Listener, and it will listen on one of the UDP ports in the range set up in the previous section. For simplicity we can use the value 500x where x will be the INPUT number. For example, 5001 for INPUT 1. Let's look at each field of the SRT protocol separately:

Mode: in the server it will always be a Listener, unless we want to connect to another server that has an SRT Listener, in which case we will have to do it through a Caller.
Address: in the case of using an SRT Listener, we will leave it blank, because it will be ignored since the server will listen to that port in all its network interfaces. In the case of using an SRT Caller, we will have to set the IP of the equipment that is listening behind a SRT Listener.

Port: is the UDP port on which we listen as SRT Listener, or on which another server listens to us if we work as an SRT caller.
Max. Latency: is the maximum latency we want in this end to end link. It can be a value between 20 and 8000 milliseconds. We will explain it in more detail later on.
Key Length: in case we want to encrypt the communication between both ends, we will have to set the length of the AES encryption (16, 24 or 32), in addition to the rest of the parameters that follow in both points.
Passphrase: this is a password, between 10 and 79 characters, to use in the AES encryption at both ends.
Encrypted: we will have to activate it, if we want to encrypt the communication between both ends.

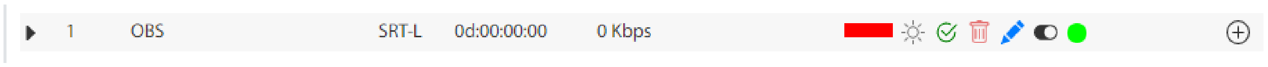
We are going to create an INPUT where OBS will connect thru SRT. To do this, we will click on ADD INPUT in the server panel, and fill in the values in this way:


NEW INPUT

Protocol	SRT
Name	OBS
Provider	Provider
Location	Estudio
Remote	Remote
Wtdg Timeout	2
Watchdog	✓
Mode	Listener
Address	
Port	5001
Max. Latency	120
Key Length	
Passphrase	**
Encrypted	

In Name we set OBS to identify the origin of the video for this INPUT, though we could have set any other descriptive name that helps us identify it easily in the panel.

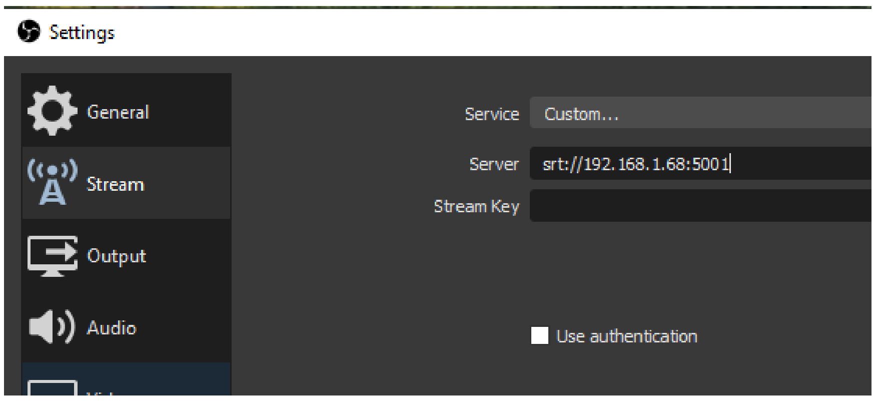
We will leave Provider blank. Location set to Studio, though we could leave it blank, since it is optional. Wtdg Timeout set it to 2 seconds, and activate the Watchdog for audio silence detection. For simplicity we set the Mode to Listener, there is no need to set the Address as explained above. We will listen on UDP port 5001, using the 500x criterion for INPUT where x is the port number. In Max. Latency we leave the default value of 120 ms, which we will evaluate in more detail later. And we are not going to encrypt the connection for simplicity.
Let’s click on Save, and something like this will appear in the panel.


1 OBS SRT-L 0d:00:00:00 0 Kbps

From left to right we will describe each visual element in detail.

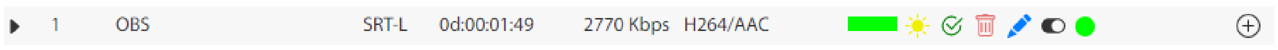
The black triangle on the left, is used to expand or collapse the OUTPUTs that copy this INPUT. This will be more clear later when we have created at least one OUTPUT. Then follows the INPUT number, which in this case is 1, since it is the first one. Then the protocol that the server will use for this INPUT, which in our case is SRT-L as an SRT Listener. Then the connection time from the latest interruption, now is currently 0, since there is no transmission yet. It is followed by the bitrate in Kbps. Then a colored bar indicating the connection status. Red means that there is no connection, green means that the connection is good, and intermediate colors are yellow to orange, if it is of low or very low quality. By hovering the pointer over this bar, a pop-up window is displayed with informative data about the connection and/or the source. Then there is a light bulb that will be off if there is no connection, or on if it’s ok. Then a checked button that allows us to deactivate this INPUT, so that even if there is a connection, it will be interrupted until we click it on again to re-establish it. Then an icon to delete this INPUT forever. If we click it by mistake, we can still click Cancel in the pop-up window that appears to warn us of the consequences of deleting this entry. Next comes the edit icon (a pencil), which allows us to change the values we want for this INPUT. Then a switch to activate or deactivate the audio watchdog, and next to it a led that will be black if the watchdog is deactivated, green if it is activated and no mute alarm has been triggered, or red if the mute alarm has been triggered and the sound has not yet returned. All the way to the right there is a button to add an OUTPUT to this INPUT, where to send the signal coming from this INPUT.

Let's go to OBS to send video to this Server INPUT. OBS uses a URL to describe its SRT caller connection, as do other software like VLC. (srt://IP_of_Server:port)
In our example the IP of the server was 192.168.1.68, and the Listener port we used was 5001, so it will look like this:

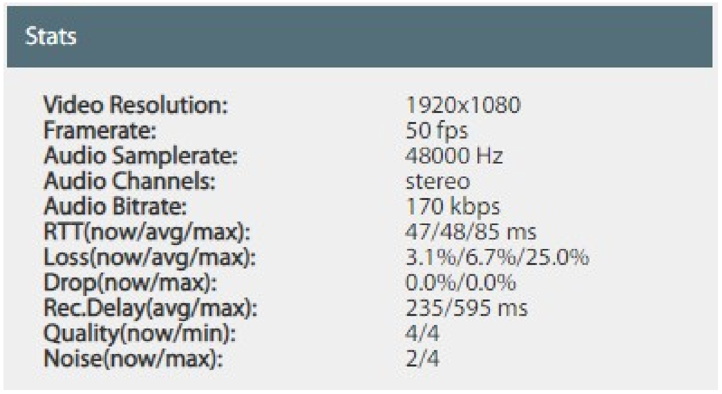

Settings

Service Custom...
Server srt://192.168.1.68:5001
Stream Key
Use authentication

When we start sending the stream, we will see something like this:


1 OBS SRT-L 0d:00:01:49 2770 Kbps H264/AAC

The input analyzer will start after a while and will recognize the video and audio codec used (H264/AAC in our example), and will detect the audio silences.
If we hover on the green bar, we will see the Stats info, that will look like this:


Stats

Video Resolution:	1920x1080
Framerate:	50 fps
Audio Samplerate:	48000 Hz
Audio Channels:	stereo
Audio Bitrate:	170 kbps
RTT(now/avg/max):	47/48/85 ms
Loss(now/avg/max):	3.1%/6.7%/25.0%
Drop(now/max):	0.0%/0.0%
Rec.Delay(avg/max):	235/595 ms
Quality(now/min):	4/4
Noise(now/max):	2/4

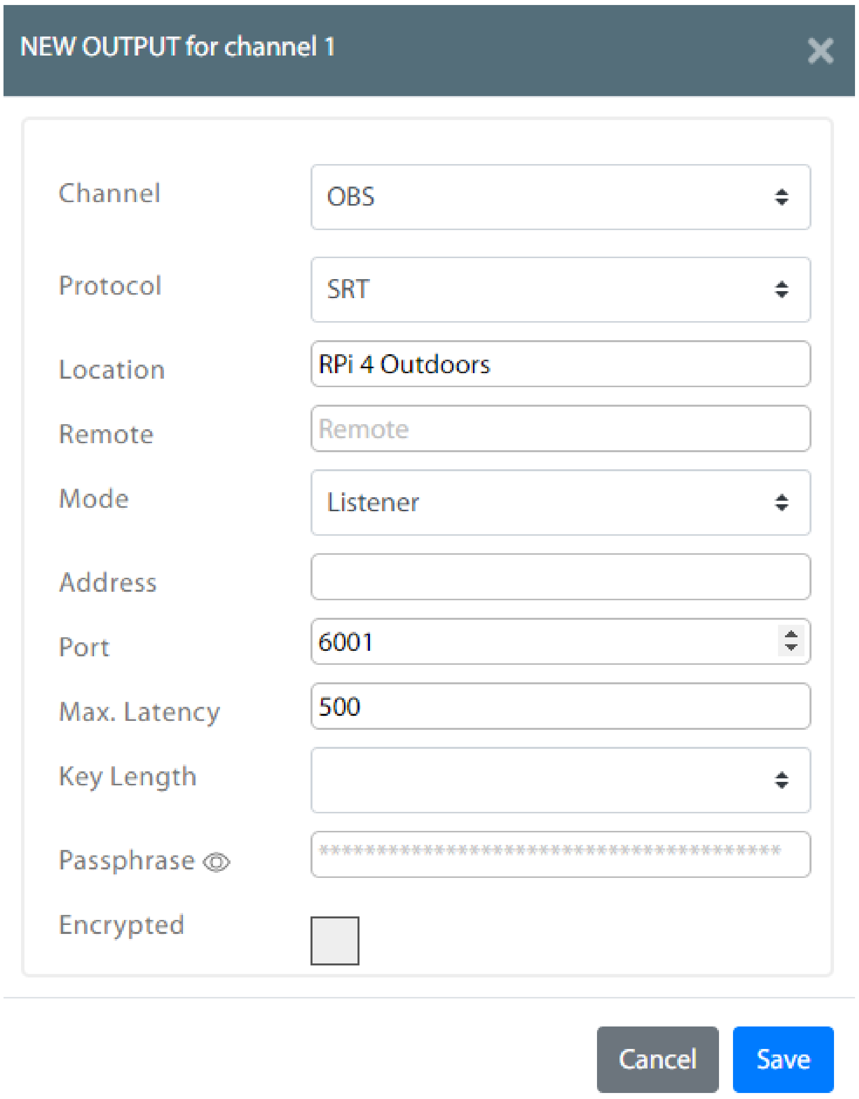
The statistical values of the SRT connection are created every second, and we explain them below:
RTT: it is the ping in the current second, the average ping in the last hour and the maximum ping in the last hour.
Loss: says the packet loss in the last second, the average in the last hour and the maximum in the last hour. This will tells us the noise in the network. Most of these lost packets will be recovered by SRT if there is enough buffer delay.
Drop: says the unrecoverable packets in the last second, and the maximum in the last hour. As you can see in this capture, in spite of and important loss number, all the packets have been recovered, and the image has not been affected with colored squares nor smear.
Rec. Delay; it is the recommended buffer delay according to the measured loss and RTT. It is recommended to set the Max. Delay of the SRT above the maximum value indicated by this parameter, if we care about quality and not about latency. In this capture a good value would be 600 ms, instead of the default 120 ms.
Quality and *Noise* are qualitative values of the image quality, from the lowest 0, to the highest 4, as well as the connection noise, from the lowest 0, to the highest 4.

Therefore, the most important value here is the maximum Rec. Delay which helps us to set the most appropriate Max. Delay value best suited to the connection. Of course, if we don't mind having a latency of about 10 seconds, and we want to be practically immune to noise, we can set it to the maximum value, that is 8000 ms (8 seconds).

Once we have an INPUT working, we can create one or several OUTPUTs creating SRT links with devices to send the input signal to, such as a repeater, or another studio, etc..

Let's create an OUTPUT to connect a Raspberry Pi Player this time. We will create an OUTPUT with an SRT Listener, and the rpiPlayer will do it as SRT Caller connecting to the IP of the server and the port where it listens. Let's assume that the rpiPlayer is outside the local network, so we will need the public IP of our Internet connection, or a DDNS domain. In our case let's assume it is 51.107.4.67 to have a concrete example.

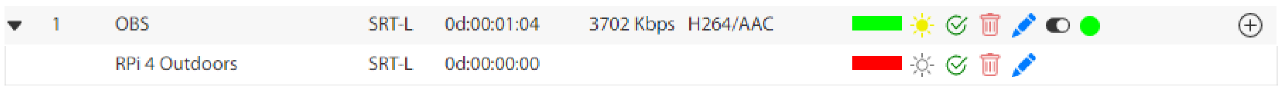
Go to the server panel, and click on the icon to the right of the INPUT OBS created earlier, to add a new OUTPUT to this INPUT.
A window opens with fields that we have already described above. We recommend to fill in at least Location, and the protocol as SRT.



NEW OUTPUT for channel 1

Channel	OBS
Protocol	SRT
Location	RPi 4 Outdoors
Remote	Remote
Mode	Listener
Address	
Port	6001
Max. Latency	500
Key Length	
Passphrase	**
Encrypted	☐

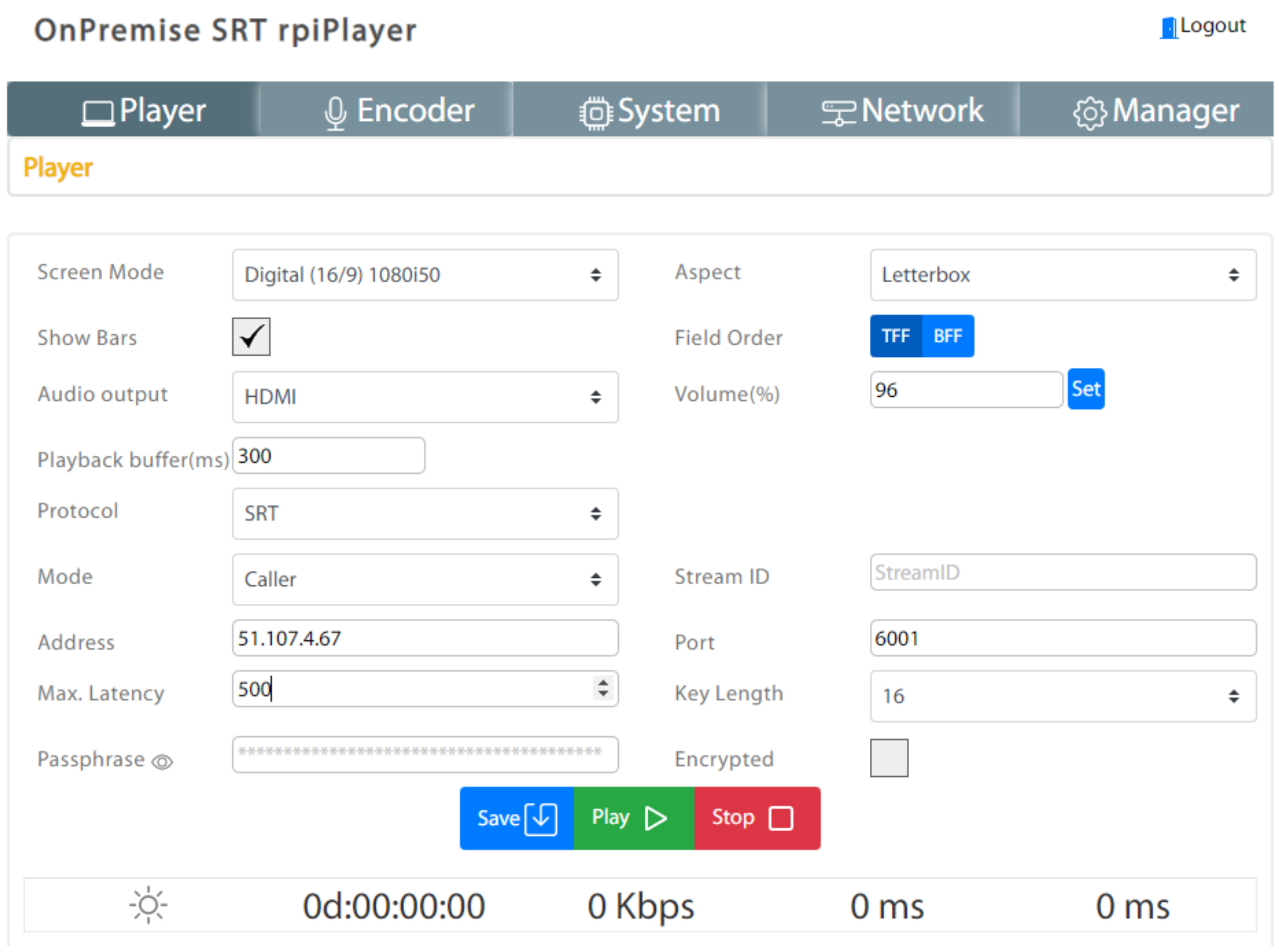
We will only comment on the Port value, which to simplify we used 600x for the OUTPUT numbered incrementally as we create them (though this is not mandatory). If by chance we use a port that is already being used in another Listener connection in the server, the system will warn us to change it to another one, so there is nothing to worry about. Click on Save, and we will have to click on the black triangle icon of the INPUT to see the new created OUTPUT.



▼	1	OBS	SRT-L	0d:00:01:04	3702 Kbps	H264/AAC	🟩
		RPi 4 Outdoors	SRT-L	0d:00:00:00			🟥

We can see components very similar to those of the INPUT, except for the bitrate or the audio and video codecs that are not shown, since when doing exact copies of the INPUT in the OUTPUT, they are exactly the same. The audio watchdog is not shown either for the same reason.

Let's now set the rpiPlayer up, which will act as an SRT Caller to connect to the OUTPUT recently created in the server side.



Screen Mode: Digital (16/9) 1080i50
Aspect: Letterbox
Show Bars: ✓
Field Order: TFF BFF
Audio output: HDMI
Volume(%): 96 Set
Playback buffer(ms): 300
Protocol: SRT
Mode: Caller
Stream ID: StreamID
Address: 51.107.4.67
Port: 6001
Max. Latency: 500
Key Length: 16
Passphrase: **
Encrypted:

0d:00:00:00 0 Kbps 0 ms 0 ms

Choose SRT in Protocol, Mode in Caller, in Address we put the public IP of the server seen from the Internet, and the port on which the OUTPUT will listen to. The latency does not have to be the same on both sides of the connection. In fact SRT will set the higher of the two ends values. A strategy is to make the server be in charge of this value, for which we set in the external device the minimum limit value we want, such as 120 ms. In this way, the delay will be at least 120 ms, or any higher value set in the server side.

The only comment specific to rpiPlayer and Vmix, is the playback buffer value, which sets the buffer that the player will take to avoid jerks if the data flow varies due to jittering when ping changes in the network. A safe value for video is 300 ms, and 1000 ms for only-audio. VLC by default uses 1000 ms for all of them.

As for the Address, which was our public IP on the Internet, we could also have set our DDNS host, in the case our public IP is not an static one.

Just like the INPUT, each OUTPUT will yield its own statistics in its SRT connection that will allow us to adjust the best suited value of the Max. Delay for our use case.

You can add as many OUTPUTs as you want to the same INPUT, and you can create as many INPUTs as the server hardware can handle. By means of the controls we can edit their values, activate or deactivate them as we wish, and delete the ones we are not going to use anymore.

However, it is advisable to deactivate it instead of deleting it, because once an element is deleted, it will have to be recreated from the beginning, and it is much easier to edit a pre-existing one than to create a new one, but of course that is up to the user.

It is also obvious that if you delete an INPUT with several OUTPUTs, they are deleted at the same time with it. By editing an OUTPUT you can also change which INPUT it copies the signal from, so that it is not necessary to delete it from one INPUT and create it again in another, but merely edit it and change its Channel or INPUT. In this easy way, we can change the content that gets to any OUTPUT in this quick and easy way.

In Youtube, we have posted some example videos, where different software are set up thru SRT, in the playlist: https://www.youtube.com/watch?v=W3jWg61VX58&list=PLDyk9LJ27Vpprw-fjRn2uBJ8WUPGIHAH2 .

To finish with the SRT links, let's deal with the StreamID field, which we have always left blank in Caller mode. This field will only be needed if the connecting server uses it to identify a specific stream. In our servers we will never use it, since we do not multiplex several streams on the same UDP port, however we added this field to be compatible with servers that do it, and it will be the administrator of that server who will have to indicate the StreamID to connect to.

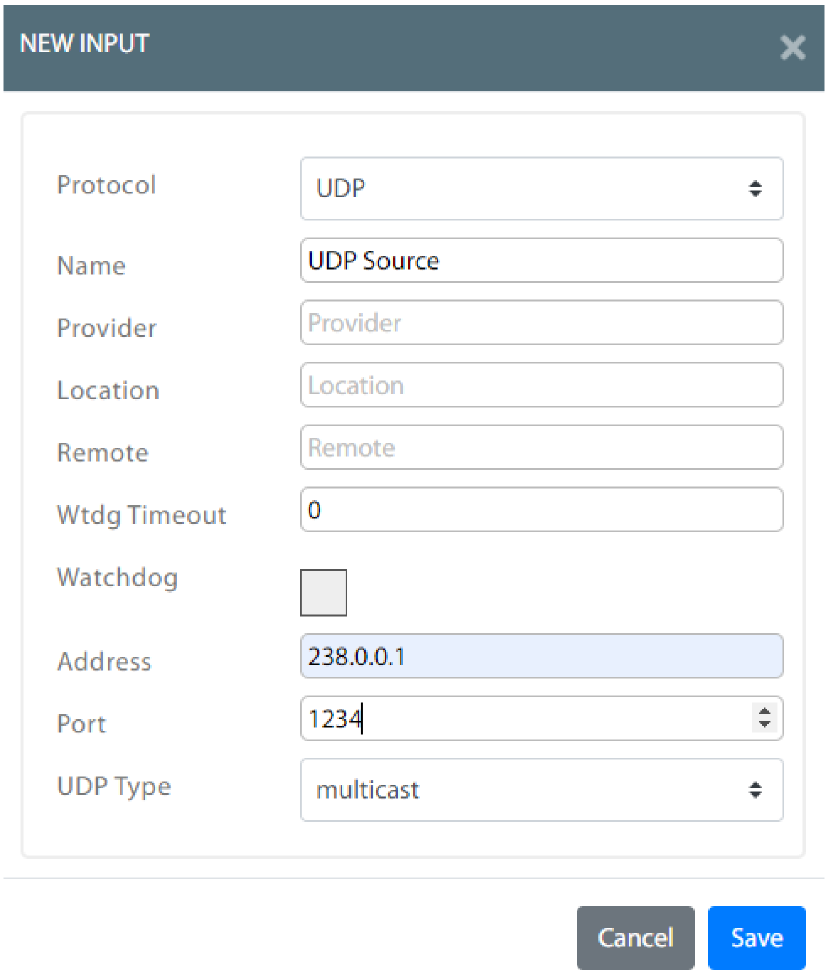
Next we will discuss UDP protocol links, also known as MPEG/IP or MPEG-TS.

UDP links

UDP is a transport protocol that by itself does not have a mechanism to recover lost packets, so we have only included it for very specific use cases.

In an INPUT the UDP protocol can receive a signal from a unicast or multicast source. In the case of a unicast source, it is like an SRT Listener, which does not require Address, since it will listen to that port on all the server interfaces. In the case of multicast, the IP address will be of the multicast type (i.e. from 224.0.0.0.0 to 239.255.255.255).

Normally the classic encoders with UDP output, do it in multicast, for example on addresses of type 238.0.0.1 on port 1234. Such an INPUT, would be created with these values.



Protocol	UDP
Name	UDP Source
Provider	Provider
Location	Location
Remote	Remote
Wtdg Timeout	0
Watchdog	☐
Address	238.0.0.1
Port	1234
UDP Type	multicast

We will not explain again the common fields from Name to Watchdog, already seen in the SRT links, and which are common to all protocols.

The UDP OUTPUTs will mainly be in multicast, for contribution in IP headends, or classic decoders that usually only carry IP input in UDP. In the OUTPUT there is only an additional field, called TTL, which tells us the number of routers through which we want it to be able to pass thru at most. The multicast network administrator can set this value, but it must be at least 1 for the stream to be sent to the local network.

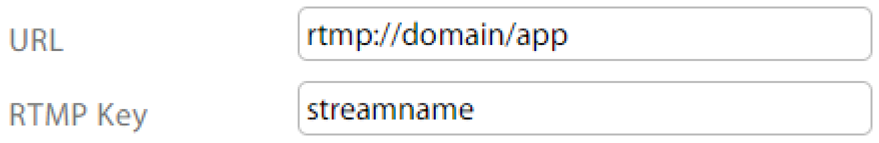
Both INPUT and OUTPUT in UDP have zero latency in our server.

RTMP protocol

The RTMP protocol was created by Adobe to be used for video streaming in conjunction with Flash technology. Today, Adobe has abandoned Flash, and it is no longer supported. The only reason RTMP is still used today is that it is still implemented as a contribution in many web streaming servers, but it is expected to be replaced by SRT over time. The RTMP standard only supports VP8 and H.264 video codecs, which limits its professional use to H.264 only. Because it is transported by TCP, its sending throughput is lowered as the RTT (ping) increases or when the packet loss rate increases, even dropping below the 800 kbps, where professional video streaming is impossible. It may be the case, where a connection in RTMP can not exceed 200 kbps, due to pings over 500 ms, and losses above 2%, and SRT is able to send 8000 Kbps without any problem at all.

However, to ease the addition of streams from servers that only serve this protocol, we have added RTMP pull to INPUTs, and RTMP push to OUTPUTs to allow publishing to services that still use it such as Youtube, Facebook, Twitch and others.

The URL of an RTMP stream always starts with rtmp:// or rtmps:// :
rtmp://domain/app/streamname
The domain part, can be a complete domain, or the IP of the RTMP server. The streamname part in some servers is known as the RTMP Key. And therefore the previous part is called URL or FMS URL.

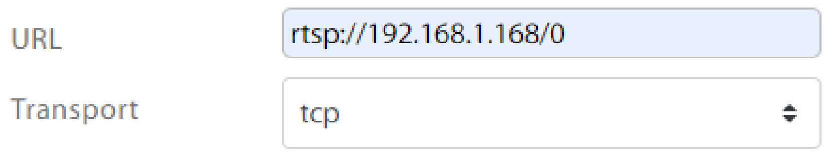


URL	rtmp://domain/app
RTMP Key	streamname

This would be the way to enter the URL of an RTMP stream, both in an INPUT and in an OUTPUT.

RTSP protocol

It is a protocol very little used nowadays, except in IP surveillance cameras, and some encoders. For this reason we have added the option to pull this type of servers only in INPUTs. An rtsp URL, goes all complete as for example: rtsp://192.168.1.168/0, and although the most typical transport layer is TCP, it can also be UDP or HTTP. This would be something that the RTSP server in question should have to provide.



URL: rtsp://192.168.1.168/0

Transport: tcp

HTTP Protocol

It is a protocol that we already know to access any website, either without on the same line encryption HTTP, or with SSL encryption, in the form HTTPS. All these URLs start with http:// or https:// .

We have included 3 types of INPUT accessible from HTTP, such as:

- HTTP progressive : typical of services such as Shoutcast or Icecast, for only-audio.
- HLS: protocol created by Apple, and today widely used on the web. Its URLs end with .m3u8
- DASH: open protocol, widely used in VoD Its URLs end with .mpd

OnPremise SRT Server Administration

You can access to OnPremise SRT Server, from 2 distinct roles: admin and user.

The admin can access the complete panel with all functions, while the user can only access the Streams and System sections.

From the Manager section, the Admin can change his password and his password recovery email, in case he forgets it. He can also set the user name and password of the server user, and also block the user so that all streams are blocked.

The administrator can also update the server when an update is available. He can also make a backup of the entire server when, for example, the entire distribution network is up and running. The file that is generated has the date and time of creation in its name, and it is of JSON type. If at any time you want to restore the system to a specific date that you saved, a Restore of this saved file can be done.

IMPORTANT: *Every time you update the software, once the server has restarted (3 seconds) and you are at the login page, use CTRL-F5 once or twice to reload the new front-end webapp in your browser. If you don't, you will be using the older version loaded in the cache of your browser.*

The License section contains information about the current license and its expiration date. It is important to save the License identifier or LICID in case we need to change the server hardware, and we want to keep our current license.

From the same panel, you can make purchases and subscriptions to the different licenses available, as well as their payment.

When the software is installed for the first time on a new hardware, a new LICID is generated with a free LICA-G license for the next 30 days. If you want to install a license from another hardware on this new computer, just enter the LICID of that license and click on Save.

The access as Admin, by default, is admin in the username, and admin in the password. If you have changed the password, and you forgot it, you can recover it by entering the recovery e-mail in the Login by clicking on the "I've forgot my login info" button.

The default user access, is user in the username, and user in the password. If you have changed the username and/or password, and you forgot it, you will be able to recover it in the same way admin does.

If you have any questions regarding the use of OnPremise SRT Server, you can contact us thru the form in our website: https://todostreaming.eu/ , we will be happy to assist you.

You can also access the training videos on Youtube from the same web page.

2021 TodoStreaming