SERVUS

Ingest and Playout Server – Introduction

About

This document describes the concepts of x-dream-media's SERVUS Server.

What it is

SERVUS – (latin "Slave") is a software-only videoserver that supports a very wide range of IO-Devices as well as IP Stream and File formats as inputs and outputs. It combines the functionality from traditionally distinct products such as Decoder, Recorder, Transcoder, Playout server, Graphics server, Encoder within one engine. Due to its nature it can be deployed on premise, in a remote datacenter or at a public cloud. Therefore it addresses traditional as well as most modern business cases.

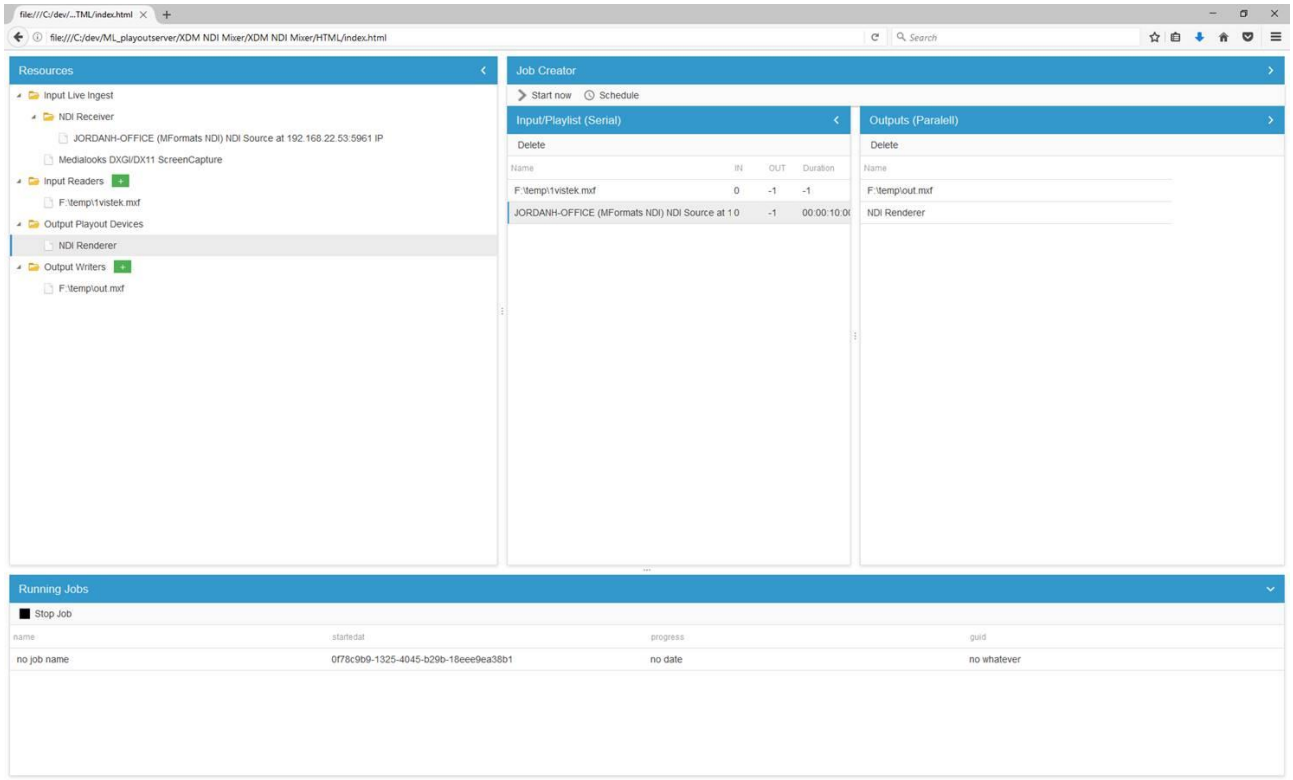

Figure 1: SERVUS Administration GUI

Supported **inputs** and **outputs** are:

- Devices
  - Video IO Cards: Blackmagic Design, AJA, BlueFish444, Stream Labs, DELTACAST, Magewell and DekTec
  - NDI (Network Device Interface developed by Newtek – SDI over IP) Sender and Receiver
  - Screen Capture, Webcam and other system devices available as Directshow Source (if started as application instead of as service)

- File/Stream Formats
  - Mostly any known Container and Codec Format (Video and Audio as well as Image) is supported for decoding and encoding.
  - Streaming protocols like http/https, udp, rtsp, rtmp are supported as input and output.
  - Special support for youtube URL's: When a link to youtube is provided as source, the highest quality of the main video is automatically parsed from youtube

Various **audio/video processing** functions are provided:

- Logo insertion
- Lower thirds, e.g. crawls, animations, graphics (future relase)
- Video up/down scaling, deinterlacing, standard conversion
- Conversion between any supported input and any output format

What to use it for

Due to it's flexibility SERVUS addresses many traditional and most modern use cases. Every user may have his own ideas and use cases. Some of the obvious and tested use cases are:

- Live feed recording
- Live stream recording
- Tape ingest
- Studio playout/play-in
- Stream playout
- Graphic overlaying
- Format conversion
- Stream switching

As explained SERVUS is intended to act as a slave under control of a controlling application. Such application may either expose its functionality for manual or for fully automatic operations.

Topology

SERVUS is designed but not limited to be part of a larger ecosystem where a governing external Controller sends remote control commands to its API.

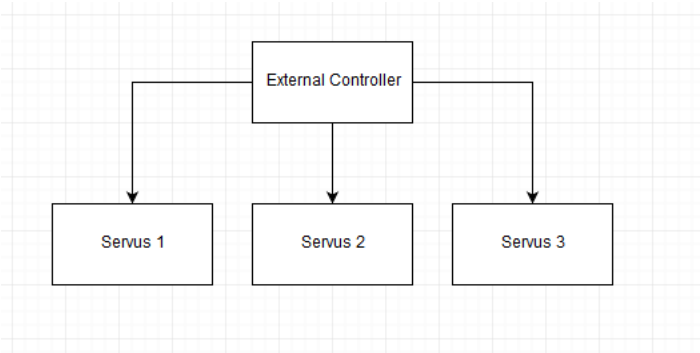

Figure 2: Topology

Each SERVUS Node can make use of multiple inputs and outputs. The limiting factor is the given computer hardware environment.

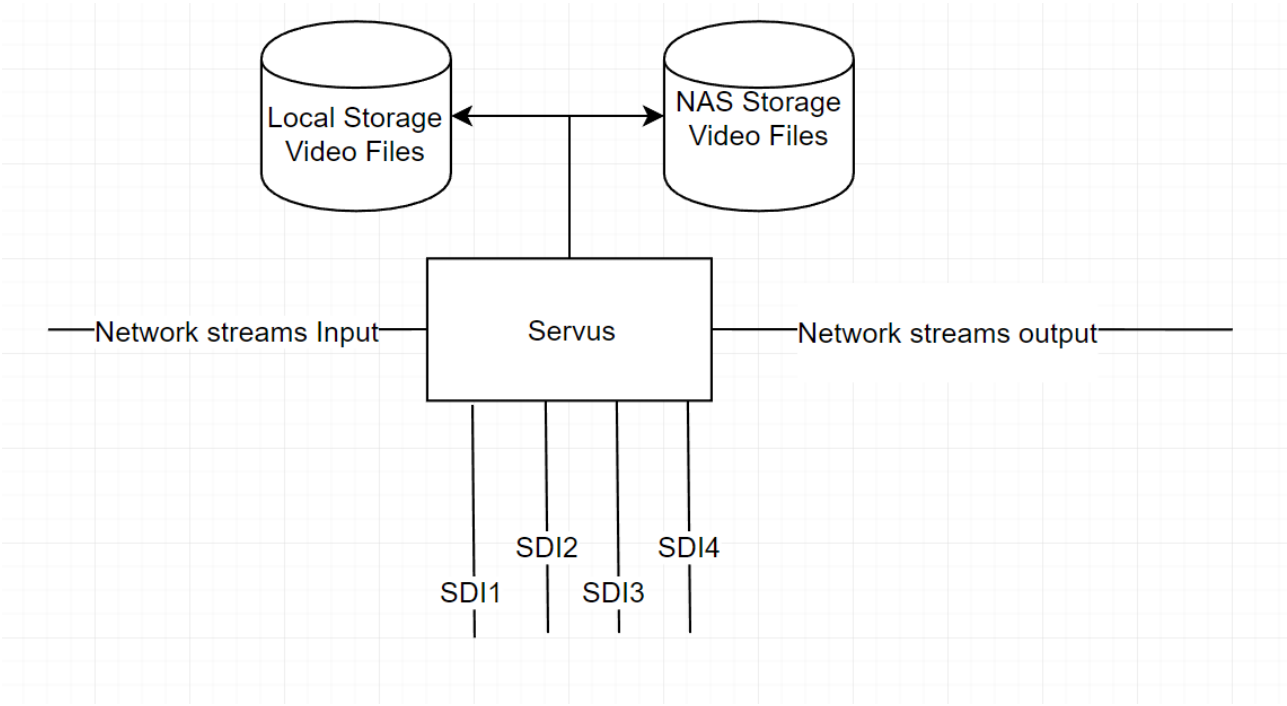

Figure 3: Inputs and Outputs

Please note: A SERVUS Node is not designed to provide any resource management. A SERVUS (Slave) will try to do its best to follow its orders even if that means it exceeds given hardware ressources. Incorrect job ececution can occur in such cases.

Job Concept

Various combinations of inputs and outputs are possible. A Job basically consists of a list of inputs and outputs.

Inputs:

In case there is more than one input for a Job, the inputs are considered to be a **playlist.** You can mix any supported input source (see first page) within the playlist. For instance the playlist could start for 10:30 with video from a capture card, then a network live stream follows for 00:30, followed by a file until its end.

Outputs:

If there is more than one output for a Job, all outputs are being served in parallel and synced to a random live output. For instance one live stream is captured to a file while it is played out at the same time to SDI and also a WebRTC preview is generated to be shown at a computer monitor (e.g. embedded within control application user interface).

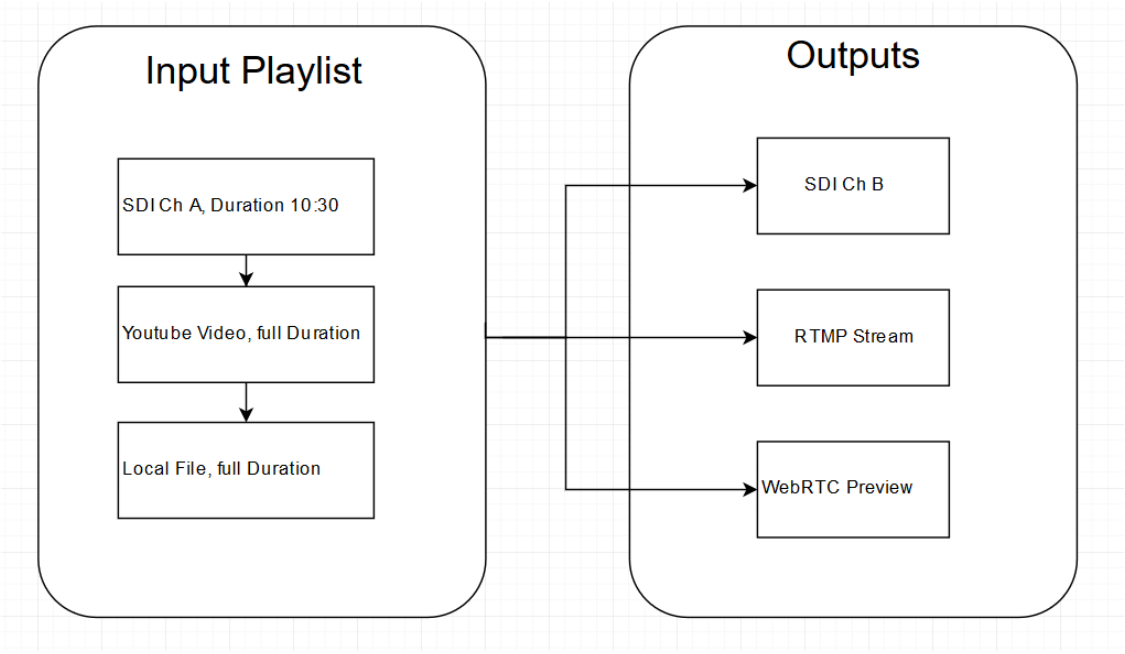

Figure 4: Example Input Playlist and Output configuration

Preview Concept

Preview can be granted into any running Job using multiple different techniques e.g. add one low-resolution UDP stream output to the Job outputs.

However, the graphical user interfaces that are provided with SERVUS are making use of a technique called WebRTC to make previews of video and audio inputs available.

Any Job where a preview on a computer monitor is requested just needs to have the “WebRTC” device added to its outputs list. When a WebRTC enabled Job starts, it calculates all details a client application needs to connect and stores it in its own JobInfo struct. A client that wants to display a preview has to know the JobGuid and the WebRTC Signaling Servers address. It can get such from the “getRunningJobs” API call.

Only when a client application subscribes or better activates the WebRTC preview, the Job really encodes the necessary data stream. When there is no more client subscribed to the signal, the encoding is just not done.

Be aware that using WebRTC is considered as a Live Output. When a file to file transcoding job runs in full speed and a client connects to the WebRTC preview of the Job, it will slow down to realtime until the client disconnects again. Also be aware that multiple WebRTC consumers on the same channel cost CPU resources on a per-client base.

About SimpleWebRTC and the Signaling Server:

SimpleWebRTC is basically a javascript wrapper that makes consuming WebRTC media streams very comfortable. There is a so called Signaling Server bundled with SERVUS that is made for a player application to connect to the desired WebRTC stream easily. Bundled as well is a special version of the simplewebrtc.js extension.

To correctly consume the WebRTC streams the same SimpleWebRTC version should be used as the current SERVUS web interface client does use. Otherwise the Signaling Server and the client framework would not work together.

Job Control

A running job can be modified at runtime. Currently SERVUS supports these commands to modify a running Job:

- add or remove items from the playlist
- switch to a specific playlist item
- pause a Job