Mcaster Manual

Professional modular system for TV signal processing

Flussonic

Table of contents

1. Products

1.1 Mcaster - Professional TV Signal Processing System

1.1.1 Overview

Mcaster is a professional solution for processing television signals, proven by 10 years of deployment experience in various broadcasting environments. The system represents a comprehensive solution for receiving, processing, transcoding, and distributing TV content with the highest level of reliability and quality.

1.1.2 Modular Architecture

Mcaster is built on the principles of modular architecture, which provides configuration flexibility and scalability for specific tasks. The system consists of specialized components, each responsible for a specific signal processing function:

Main Modules

SIGNAL CAPTURE AND RECEPTION

- **SDI Coder** — video and audio capture through SDI cards (Dektec, Blackmagic, V4L)
- **ASI Reader** — ASI flow reception with Dektec, Streamlabs, Softlab support
- **SRT Reader** — SRT protocol publication reception with minimal delay

PROCESSING AND MANAGEMENT

- **LiveStreamInput (LSI)** — automatic switching between primary and backup sources
- **SCTE Processor** — SCTE35/SCTE104 advertising marker processing with automatic compensation
- **DVB-WebVTT** — DVB subtitle recognition and conversion to WebVTT

TRANSCODING AND ADAPTATION

- **Transcoder** — transcoding for DVB (CBR) and OTT (MBR) with H.264/H.265 support
- **Multiplexer** — transport flow formation
- **Format adapters** — conversion between various broadcasting standards

Modular Architecture Advantages

- **Configuration flexibility** — selection of only necessary components
- **Scalability** — adding modules as needs grow
- **Reliability** — function isolation, reduced failure risk
- **Easy maintenance** — independent module updates and diagnostics
- **Specialization** — optimization of each module for specific tasks

1.1.3 Built-in Monitoring

A special advantage of Mcaster is the built-in monitoring system, which provides detailed analysis of incoming flows at a level unavailable to other solutions on the market.

Monitoring Capabilities

FLOW ANALYSIS

- **Detailed statistics** of each module in real time

- **Signal quality metrics** with frame-level accuracy
- **Bitrate analysis** and encoding parameters
- **Delay monitoring** and synchronization

SPECIALIZED METRICS

- **SRT parameters** — RTT, latency, retransmitted_packets
- **LSI statistics** — source switching, time on backups
- **SCTE metrics** — advertising marker processing
- **OCR quality** — subtitle recognition accuracy

DIAGNOSTICS AND ALERTS

- **Automatic problem detection** and anomalies
- **Failure prediction** potential failures
- **Detailed reporting** for trend analysis
- **Integration** with external monitoring systems

Built-in Monitoring Advantages

- **Analysis depth** — unavailable to third-party solutions
- **Real time** — instant response to changes
- **Proactivity** — preventing problems before they occur
- **Optimization** — continuous broadcasting quality improvement

1.1.4 Deployment Options

Mcaster is available in three different deployment options, allowing you to choose the optimal solution for any infrastructure:

1. Standalone Linux Program

Application: Integration into existing infrastructure - **Flexibility** — installation on any compatible system - **Cost-effectiveness** — use of existing equipment - **Integration** — work with existing monitoring systems - **Scalability** — clustering capability

2. Server Firmware

Application: Turnkey solution - **Easy deployment** — automatic OS and software installation - **Optimization** — system configured for specific tasks - **Security** — isolated execution environment - **Reliability** — proven configuration

3. Hardware-Software Complex

Application: Maximum reliability and performance - **Refined equipment** — tested servers and capture cards - **Guaranteed compatibility** — all components tested - **Technical support** — comprehensive maintenance - **Performance** — configuration optimized for tasks

1.1.5 Application Areas

Television Broadcasting

- **Terrestrial broadcasting** — flow preparation for DVB-T/T2/S
- **Cable TV** — signal processing for cable networks
- **Satellite broadcasting** — content adaptation for satellite channels

OTT and Internet Broadcasting

- **Adaptive flows** — HLS, DASH for various devices
- **Multi-bitrate** — optimization for connection quality
- **Global distribution** — content delivery worldwide

Professional Applications

- **Studio broadcasting** — real-time signal processing
- **Mobile applications** — streaming for smartphones and tablets
- **Corporate networks** — internal broadcasting in organizations

1.1.6 Mcaster Advantages

Reliability

- **10-year experience** of deployments in various conditions
- **Proven architecture** — thousands of successful installations
- **Fault tolerance** — automatic switching to backups
- **Stability** — continuous operation in critical conditions

Quality

- **Professional algorithms** for signal processing
- **High accuracy** of synchronization and encoding
- **Optimization** for various content types
- **Compatibility** with international standards

Efficiency

- **Modular architecture** — use of only necessary components
- **Optimized performance** — minimal resource consumption
- **Automation** — reduced manual intervention
- **Scalability** — growth with needs

1.1.7 Conclusion

Mcaster represents a professional solution for processing television signals, combining proven reliability, modular architecture, and unique monitoring capabilities. 10 years of deployment experience and thousands of successful installations make it the choice of broadcasting professionals.

The flexibility of deployment options allows adapting the solution to any infrastructure, while built-in monitoring provides quality control at a level unavailable to competitive solutions. Mcaster is not just software, it is a comprehensive solution for professional broadcasting.

2. Manual

2.1 Installing mcaster appliance

2.1.1 Installation Process Overview

Installing mcaster in appliance mode is a fully automated process that requires no human intervention after initial setup. We provide a ready-made filesystem image that contains the installer and all necessary components for system deployment.

Mcaster can be installed in two ways: as our software appliance that runs on standard server hardware, or by reinstalling firmware on our PAK (Professional Appliance Kit) dedicated hardware. The appliance installation involves creating a bootable media and following an automated installation process, while PAK installation requires downloading and installing the latest firmware through the PAK management interface.

2.1.2 Obtaining the Installer Image

Requesting the Image

To obtain the installer image, you need to contact our technical support. Upon request, we provide:

- **Filesystem image link** - ready ISO image for USB recording
- **Recording instructions** - recommendations for creating a bootable USB drive
- **Configuration documentation** - description of configuration parameters

USB Drive Requirements

- **Capacity**: minimum 4 GB (8 GB recommended)
- **Format**: FAT32 or exFAT
- **Speed**: USB 3.0 or higher

2.1.3 Installation Preparation

Recording the Image to USB

1. **Download the image** - download the provided ISO file
2. **Record the image** - use specialized utilities:
3. **Windows**: Rufus, Win32 Disk Imager
4. **macOS**: Etcher, Disk Utility
5. **Linux**: dd, Etcher

```
# Example of recording image in Linux
sudo dd if=mcaster-installer.iso of=/dev/sdX bs=4M status=progress
```

Configuring Installation Parameters

Before installation, you can configure the parameters of the future appliance by editing the `autoinstall.txt` file on the USB drive.

2.1.4 autoinstall.txt Configuration File

File Structure

The `autoinstall.txt` file contains configuration variables in `KEY=VALUE` format:

```
# License key
LICENSE=your-license-key-here
```

```
# Administrator credentials
EDIT_AUTH=admin:securepassword

# API key for centralized management
CENTRAL_API_KEY=your-api-key-here

# Configuration partition size (in MB)
SIZE_SETTINGS=200

# Log partition size (in MB)
SIZE_VAR=4096
```

Configuration Parameters

Parameter	Description	Required	Example
`LICENSE`	License key for mcaster	Yes	`LICENSE=MC-XXXX-XXXX-XXXX`
`EDIT_AUTH`	Administrator login and password	Yes	`EDIT_AUTH=admin:mypassword`
`CENTRAL_API_KEY`	API key for centralized management	No	`CENTRAL_API_KEY=api-key-123`
`SIZE_SETTINGS`	Size of `/etc` partition in MB	No	`SIZE_SETTINGS=200`
`SIZE_VAR`	Size of `/var` partition in MB	No	`SIZE_VAR=4096`

Recommended Settings

```
# Minimal configuration
LICENSE=your-license-key
EDIT_AUTH=admin:complex-password-123

# Extended configuration
LICENSE=your-license-key
EDIT_AUTH=admin:complex-password-123
CENTRAL_API_KEY=your-central-api-key
SIZE_SETTINGS=500
SIZE_VAR=8192
```

2.1.5 Installation Process

Booting from USB

1. **Insert USB drive** into the server
2. **Configure BIOS/UEFI** to boot from USB (usually F2, F12, or Del)
3. **Select USB drive** in boot menu
4. **Wait for installer** to load

Automatic Installation

After booting from USB, a fully automatic installation occurs:

1. **Initialization** - loading installer into memory
2. **Hardware scanning** - determining server configuration
3. **Disk erasure** - complete cleaning of system hard drive
4. **Partition layout** - creating partitions according to settings
5. **System installation** - copying filesystem images
6. **Configuration setup** - applying parameters from `autoinstall.txt`
7. **Reboot** - automatic restart into new system

Timeframes

- **Installer loading**: 1-2 minutes

- **Disk erasure**: 5-15 minutes (depends on disk size and speed)
- **System installation**: 10-20 minutes
- **Total time**: 15-40 minutes

2.1.6 Security and Recommendations

⚠ Important Warning

WARNING: Booting from USB drive on a production server will result in complete data erasure and system reinstallation. This is not considered a fatal problem since production server configurations should be entered into a centralized management system.

Recommended Secure Method

For maximum security, it is recommended to specify in `autoinstall.txt`:

- **License key** (`LICENSE`)
- **Administrator credentials** (`EDIT_AUTH`)

This ensures:

- **Automatic activation** - server doesn't remain on network waiting for license input
- **Immediate availability** - system is ready to work immediately after installation
- **Security** - elimination of manual input of sensitive data

Installation Preparation

1. **Backup** - save important data from server
2. **Document configuration** - record current settings
3. **Check compatibility** - ensure compliance with system requirements
4. **Prepare network** - configure network connection for update downloads

2.1.7 Post-Installation Setup

First Boot

After installation, the system automatically:

- **Applies license** - activates mcaster
- **Configures network** - obtains IP address via DHCP
- **Creates administrator** - sets up credentials
- **Connects to centralized management** (if API key is specified)

System Access

- **Web interface**: `http://server-IP-address`
- **SSH access**: `ssh admin@server-IP-address`
- **Login/password**: specified in `EDIT_AUTH`

Installation Verification

1. **Check web interface** - accessibility of control panel
2. **Check license** - activation status in system
3. **Check logs** - absence of critical errors

4. **Check network** - accessibility of external resources

2.1.8 Troubleshooting

Common Issues

Problem	Solution
Won't boot from USB	Check BIOS/UEFI settings
Image recording error	Use different USB drive
Can't read autoinstall.txt	Check file format (UTF-8)
License error	Check key correctness
Network issues	Check cables and DHCP settings

Installation Logs

In case of installation problems, logs are saved in:

- **USB drive**: `/logs/install.log`
- **Screen**: display of installation process
- **Network**: sending logs to central server (if configured)

2.1.9 Conclusion

Installing mcaster appliance is a simple and reliable process that ensures rapid system deployment with minimal human intervention. Automatic installation guarantees configuration reproducibility and reduces the likelihood of deployment errors.

2.2 Appliance Operating System

2.2.1 Architecture Overview

Our infrastructure media appliance uses a specially developed operating system built on Linux with a number of key modifications to ensure maximum security, reliability, and efficiency.

InfraMedia OS is provided as the base operating system in our appliance and PAK (Professional Appliance Kit) hardware deliveries. This specialized OS is pre-configured and optimized for media server operations, ensuring consistent performance and security across all deployment scenarios.

2.2.2 Minimal Package Set

Minimalism Principle

We independently compile a minimal set of packages for the operating system, which allows:

- **Minimize attack surface** - fewer packages mean fewer potential vulnerabilities
- **Reduce firmware size** - compact system loads faster and takes up less space
- **Improve stability** - excluding unnecessary components reduces the likelihood of conflicts
- **Simplify support** - fewer components are easier to test and update

Build Process

1. **Dependency analysis** - thorough analysis of necessary components for media server operation
2. **Source compilation** - building packages from verified source codes
3. **Security validation** - checking each package for known vulnerabilities
4. **Size optimization** - removing unnecessary files and compressing components

2.2.3 File System Images

Readonly Architecture

After installing the minimal package set, we create **readonly file system images**:

- **Immutability** - file system is protected from accidental changes
- **Integrity** - impossibility of modifying system files
- **Reproducibility** - identical images for all devices

Linux Kernel Modification

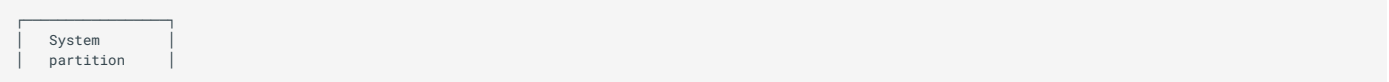
Our Linux kernel is modified to work with file system images instead of traditional installation:

- **Image support** - kernel can mount and work with FS images
- **Partition management** - special drivers for working with image partitions
- **Performance optimization** - fast loading and working with images

2.2.4 Additional Software Installation

Installation Mechanism

Additional software is installed by copying new file system images to a separate partition:



```
|  (readonly)     |
├─────────────────┤
|   User          |
|   partition     |
|   (readwrite)   |
├─────────────────┤
|   Software      |
|   partition     |
|   (FS images)   |
└─────────────────┘
```

Approach Advantages

- **Isolation** - additional software does not affect the main system
- **Versioning** - each software version is stored in a separate image
- **Rapid deployment** - copying image instead of installation
- **Change rollback** - ability to return to previous version

2.2.5 Update System

Safe Updates

System updates occur by loading a new image without overwriting the old one:

1. **Loading new image** - new image is loaded to a separate partition
2. **Validation** - checking integrity and compatibility of the new image
3. **Switching** - changing boot parameters to use the new image
4. **Testing** - checking functionality after update

Rollback Mechanism

If something goes wrong during the update, the system can automatically or manually rollback to the previous configuration:

- **Preserving previous versions** - old images remain available
- **Fast rollback** - switching to previous image takes seconds
- **Automatic recovery** - system can automatically rollback on critical errors

Update System Advantages

- **Reliability** - impossibility to "break" the system during updates
- **Minimal downtime** - fast updates and rollbacks
- **Production testing** - ability to safely test new versions
- **Version history** - preserving all previous versions for analysis

2.2.6 Security

Multi-level Protection

Our architecture provides several levels of security:

1. **Minimal attack surface** - only necessary components
2. **Readonly file system** - protection from system file modification
3. **Component isolation** - separation of system and applications
4. **Version control** - ability to quickly rollback when vulnerabilities are detected

Monitoring and Auditing

- **Logging all changes** - complete history of modifications
- **Integrity verification** - checksums for all images
- **Automatic anomaly detection** - monitoring unusual activity

2.2.7 Conclusion

Our appliance's operating system represents a highly optimized solution that provides maximum security, reliability, and efficiency for infrastructure media applications. Using file system images, minimal package sets, and secure update systems makes our solution ideal for critical production environments.

3. Modules

3.1 MPEGTS Reader

3.1.1 Overview

MPEGTS Reader is the most important module of Mcaster that allows receiving MPEG transport streams in both SPTS (Single Program Transport Stream) and MPTS (Multiple Program Transport Stream) modes. The module provides complete processing of transport streams with automatic extraction of metadata and content.

3.1.2 Operating Principles

Complete Stream Processing

When receiving, the stream is completely parsed down to frames, the entire transport container is unpacked. This is a fundamental difference from remultiplexers that leave MPEGTS and PES packaging with all their problems.

Automatic Metadata Extraction

The module automatically extracts and processes:

- **SDT (Service Description Table)** — information about channels and providers
- **EPG (Electronic Program Guide)** — program guide
- **PAT/PMT** — program tables
- **PES packets** — elementary streams

Demultiplexing

For MPTS streams, the module performs:

- **Automatic detection** of programs in the stream
- **Extraction of selected programs** by number
- **PID filtering** for processing optimization
- **Preservation of synchronization** between streams

3.1.3 Configuration

Basic SPTS Stream

The simplest configuration example for receiving SPTS:

```
stream s {
  input udp://239.0.0.1:1234;
}
```

MPTS Stream with Demultiplexing

For receiving MPTS, demultiplexing needs to be specifically requested:

```
stream s {
  input mpts-udp://239.0.0.1:1234?programs=1050;
}
```

Important: When using the `mpts-udp` protocol, the required multicast group will be received on the server only once, regardless of the number of streams using this group. This ensures efficient use of network resources and prevents traffic duplication.

Advanced Configuration

```
stream main_channel {
  input mpts-udp://239.0.0.1:1234?programs=1050,1051 pids=100,101,102;
}
```

Configuration Parameters

Parameter	Description	Required	Example
`udp://`	UDP protocol	Yes	`udp://239.0.0.1:1234`
`mpts-udp://`	MPTS protocol	Yes	`mpts-udp://239.0.0.1:1234`
`programs`	Program numbers	No	`programs=1050,1051`
`pids`	PID filter	No	`pids=100,101,102`

3.1.4 Automatic Metadata Extraction

SDT (Service Description Table)

If SDT is present in the incoming stream, the following will be automatically extracted:

- **Channel name** — display name of the program
- **Provider name** — information about the broadcasting company
- **Service type** — type of content (TV, radio, data)
- **Country and language** — regional information

EPG (Electronic Program Guide)

If EPG is present in the stream, it will automatically:

- **Be parsed** — extraction of structured data
- **Be available** for further packaging on output
- **Be provided** for reading via JSON API
- **Be cached** for fast access

3.1.5 Filtering and Optimization

PID Filtering

The `pids` option allows filtering incoming PIDs:

```
# Excluding unnecessary audio tracks
stream filtered {
  input mpts-udp://239.0.0.1:1234?programs=1050 pids=100,101,102;
}

# Excluding teletext
stream no_teletext {
  input mpts-udp://239.0.0.1:1234?programs=1050 pids=100,101,102,103;
}
```

Filtering Benefits

- **Reduced processor load**
- **Memory savings** — processing only needed streams
- **Improved quality** — excluding problematic PIDs
- **Network optimization** — transmitting only necessary content

3.1.6 Automation

Minimal Configuration

The number of various settings is maximally reduced to relieve administrators of problems. All functionality is enabled automatically:

- **Auto-detection** of stream type (SPTS/MPTS)
- **Auto-extraction** of metadata
- **Auto-parsing** of EPG data
- **Auto-optimization** of processing

Smart Processing

The module automatically:

- **Determines structure** of incoming stream
- **Selects optimal** processing parameters
- **Adapts** to changes in the stream
- **Recovers** after errors

3.1.7 API and Interfaces

3.1.8 Monitoring and Diagnostics

Key Metrics


```
{
  "stats": {
    "input": {
      "pids": [
        {
          "pid": 1040,
          "pnr": 27,
          "packets": 125000,
          "frames": 5026,
          "empty_packets": 20,
          "errors_ts_scrambled": 70,
          "errors_ts_pmt": 10,
          "errors_ts_cc": 250,
          "dts_goes_backwards": 23
        }
      ]
    }
  }
}
```


Diagnostic Parameters

BASIC STREAM PARAMETERS

- **packets_received** — number of received packets
- **packets_lost** — lost packets
- **programs_detected** — detected programs
- **sdt_found** — presence of SDT table
- **epg_found** — presence of EPG data

DETAILED PID PARAMETERS (STATS.INPUT.PIDS[0])

Basic metrics:

- **pid** — stream identifier
- **pnr** — program number
- **packets** — number of packets
- **frames** — number of frames
- **empty_packets** — packets without payload and adaptation field

Transport stream errors:

- **errors_adaptation_broken** — packets with adaptation field larger than packet size
- **errors_ts_scrambled** — amount of scrambled TS packets
- **errors_ts_pmt** — how many times PMT was not received after 0.5 seconds
- **errors_ts_cc** — how many MPEG-TS packets were received with non-contiguous continuity counters
- **errors_ts_tei** — how many MPEG-TS packets with Transport Error Indicator were received
- **errors_ts_psi_checksum** — how many times have received PSI entry with broken checksum
- **errors_pid_lost** — how many times pid has been lost

PES packet errors:

- **broken_pes_count** — how many PES packets were started not from startcode
- **broken_pes_sum** — how many bytes were discarded due to lack of PES startcode

Time corrections:

- **time_corrections** — jumps of timestamps inside a MPEG-TS stream
- **repeated_frames** — in case of CC error last frame can be repeated. This is a count of repeated frames
- **corrected_backward_pts** — how many times PTS was less than PCR or previous PTS
- **pcr_resync** — if PTS is drifting away from PCR, it can be resynchronized with PCR. This is a resync count
- **dts_goes_backwards** — time on this PID jumped back from reference PTS and it was not a roll over zero
- **dts_jump_forward** — time on this PID jumped forward too far away from reference PTS
- **too_large_dts_jump** — jump of the PTS was so big from previous, that had to flush all frames and restart parsing

Buffering and discarding:

- **discarded_buffer_count** — how many times was discarded too big ES buffer without making a frame of it
- **discarded_buffer_sum** — how many bytes were lost due to discarding ES buffer

Service data:

- **fillers_count** — how many H264(5) NAL fillers were seen in the input
- **fillers_sum** — how many bytes were seen in NAL fillers
- **padding_pes_count** — how many PES packets were on the Padding streamId
- **padding_pes_sum** — how many bytes were in PES packets on the Padding streamId

Critical errors:

- **crashed** — unhandled crashes inside mpegts decoding process

3.1.9 Usage Examples

Simple SPTS Reception

```
stream news_channel {
  input udp://239.0.0.1:1234;
}
```

MPTS with Program Selection

```
stream main_program {
  input mpts-udp://239.0.0.1:1234?programs=1050;
}

stream secondary_program {
  input mpts-udp://239.0.0.1:1234?programs=1051;
}
```

Audio Filtering

```
stream russian_audio {
  input mpts-udp://239.0.0.1:1234?programs=1050&pids=100,101;
}
```

Transcoder Integration

```
stream transcoded {
  input mpts-udp://239.0.0.1:1234?programs=1050;

  transcoder {
    video {
      codec h264;
      bitrate 5000k;
    }
    audio {
      codec aac;
      bitrate 128k;
    }
  }
}
```

3.1.10 Troubleshooting

Common Problems

STREAM NOT RECEIVING

1. **Check network settings** — multicast address availability
2. **Ensure correctness** of program number for MPTS
3. **Check format** of incoming stream
4. **Monitor module logs**

MISSING METADATA

1. **Check presence of SDT** in incoming stream
2. **Ensure correctness** of EPG data
3. **Check PID filtering** settings
4. **Wait for metadata** updates

DEMULTIPLEXING PROBLEMS

1. **Check program number** in MPTS
2. **Ensure stability** of incoming stream
3. **Check synchronization** between streams
4. **Monitor processing** statistics

Diagnostic Commands

```
# Network traffic monitoring
tcpdump -i any -n host 239.0.0.1

# Check stream metrics
curl -sS "http://localhost:8080/streamer/api/v3/streams/stream_name"
```

3.1.11 Configuration Recommendations

Performance Optimization

- **Use PID filtering** to reduce load
- **Configure buffer size** for your streams
- **Monitor processing** statistics
- **Plan redundancy** for critical streams

Reception Quality

- **Check stability** of network connection
- **Use quality** network cables
- **Monitor packet loss** in real-time
- **Configure timeouts** for your network

Working with Metadata

- **Regularly check** SDT relevance
- **Monitor EPG data** updates
- **Use caching** for fast access
- **Plan handling** of metadata errors

3.1.12 Conclusion

MPEGTS Reader is a powerful and flexible module for receiving and processing MPEG transport streams. Automatic metadata extraction, smart filtering, and minimal configuration make it an ideal solution for professional broadcasting. Support for both SPTS and MPTS modes ensures universal application in various broadcasting scenarios.

3.2 T2MI Reader

The T2MI reader module allows decapsulating DVB-T2 network signals packaged in MPEG-TS, extracting PLP (Physical Layer Pipe) from them and then processing it as regular MPTS.

3.2.1 Automatic Enable

The module is automatically enabled when using Mpegts Reader, no configuration is required for this.

3.2.2 Purpose

The module is necessary for local processing of broadcast signals, for example:

- Regional advertising insertion
- Local content modification
- Extraction of individual services from multiplex
- Analysis and monitoring of DVB-T2 streams

3.2.3 Technical Details

Detailed description of the T2MI protocol is contained in document TS 102 773, section 6.

Processing

The module performs the following operations:

1. Receiving MPEG-TS stream with DVB-T2 signal
2. Decoding T2MI (T2-MI) containers
3. Extracting PLP (Physical Layer Pipe) data
4. Forming standard MPEG-TS stream for further processing

Supported Formats

The module works with the following data types:

- DVB-T2 signals
- MPEG-TS containers
- T2MI packaged streams
- PLP (Physical Layer Pipe) data

3.2.4 Applications

T2MI reader is used in various scenarios:

- Television broadcasting
- Cable networks
- Satellite broadcasting
- Local content insertion
- Signal quality monitoring

3.3 ASI Reader

3.3.1 Overview

ASI Reader is a module in mcaster designed for receiving video from ASI (Asynchronous Serial Interface) capture cards. The module provides reliable reception of MPTS (Multi-Program Transport Stream) flows and their transmission to multicast for further processing by other system components.

3.3.2 Supported Capture Cards

Dektec

- **Full support** for all Dektec ASI models
- Automatic signal parameter detection
- Support for all ASI standards
- Integration with Dektec SDK

Streamlabs

- **Limited support** for Streamlabs cards
- Basic ASI capture functions
- V4L2 compatibility

Softlab

- **Limited support** for Softlab cards
- Basic ASI reception functions
- V4L2 compatibility

3.3.3 Architecture

Operating Principle

1. **ASI signal capture** from the card
2. **MPTS flow processing**
3. **Multicast transmission** to the specified address
4. **Capture by mpegts reader module** for further processing

Connection Diagram

```
ASI Source → ASI Reader → Multicast → MPEGTS Reader → Processing
```

Importance of Multicast Architecture

Using multicast architecture is a **necessary condition** for ensuring streamer stability and independence from ASI card status. This provides:

- **Separation of responsibilities** between capture and processing
- **Streamer independence** from ASI card state
- **Component restart capability** without interrupting the flow
- **Scalability** — multiple consumers can receive one stream
- **Fault tolerance** — when there are problems with the card, the streamer continues to work with buffered data

3.3.4 Module Configuration

Basic Configuration

```
dvb_card asi_port_1 {
  hw dektec_asi;
  video_device 224.1.1.1:1000;
  serial 2174223642;
  port 1;
}
```

Configuration Parameters

Parameter	Description	Required	Example
`hw`	Hardware type	Yes	`dektec_asi`
`video_device`	Multicast group address	Yes	`224.1.1.1:1000`
`serial`	Card serial number	Yes	`2174223642`
`port`	Port number on the card	Yes	`1`

Extended Configuration

```
dvb_card asi_port_1 {
  hw dektec_asi;
  video_device 224.1.1.1:1000;
  serial 2174223642;
  port 1;

  # Additional parameters
  buffer_size 8192;
  timeout 5000;
  retry_count 3;
}
```

3.3.5 Card Serial Number Determination

Using DtInfoCL

To determine the Dektec card serial number, use the DtInfoCL utility:

1. **Download DtInfoCL** from the official Dektec website
2. **Install the utility** on the system
3. **Run the command** to view connected devices:

```
# View all Dektec devices
dtinfocl --list-devices

# Detailed information about a specific device
dtinfocl --device 0 --info
```

DtInfoCL Output Example

```
Device 0: DTA-2174B
  Serial Number: 2174223642
  Firmware Version: 2.1.3
  Ports: 4
  Status: Ready
```

3.3.6 Integration with mpegts reader

Processing Chain Configuration

```
# ASI Reader - capture from card
dvb_card asi_port_1 {
  hw dektec_asi;
```

```
  video_device 224.1.1.1:1000;
  serial 2174223642;
  port 1;
}

# MPEGTS Reader - reception from multicast
stream asi_stream {
  input mpts-udp://224.1.1.1:1000?programs=1070;
}
```

3.3.7 Troubleshooting

Common Problems

CARD NOT DETECTED

1. **Check ASI cable connection**
2. **Verify serial number correctness**
3. **Check Dektec drivers**
4. **Use DtInfoCL for diagnostics**

NO SIGNAL IN MULTICAST

1. **Check multicast group settings**
2. **Ensure port availability**
3. **Check network settings**
4. **Check module logs**

BUFFER ERRORS

1. **Increase buffer size**
2. **Check system performance**
3. **Optimize network settings**

Diagnostic Commands

```
# Check multicast flow
tcpdump -i any -n host 224.1.1.1

# Test multicast connection
nc -u 224.1.1.1 1000

# Check Dektec device status
dtinfocl --list-devices
```

3.3.8 System Requirements

Hardware Requirements

- **CPU**: Minimum 2 cores
- **RAM**: 4 GB
- **Network**: Gigabit Ethernet for multicast
- **ASI card**: Supported Dektec/Streamlabs/Softlab model

Software Requirements

- Linux kernel 4.19+
- Dektec drivers (for Dektec cards)
- V4L2 support (for Streamlabs/Softlab)
- Multicast support in network

Network Requirements

- **Multicast**: Enabled in network infrastructure
- **IGMP**: Support for multicast management
- **Ports**: Availability of specified ports

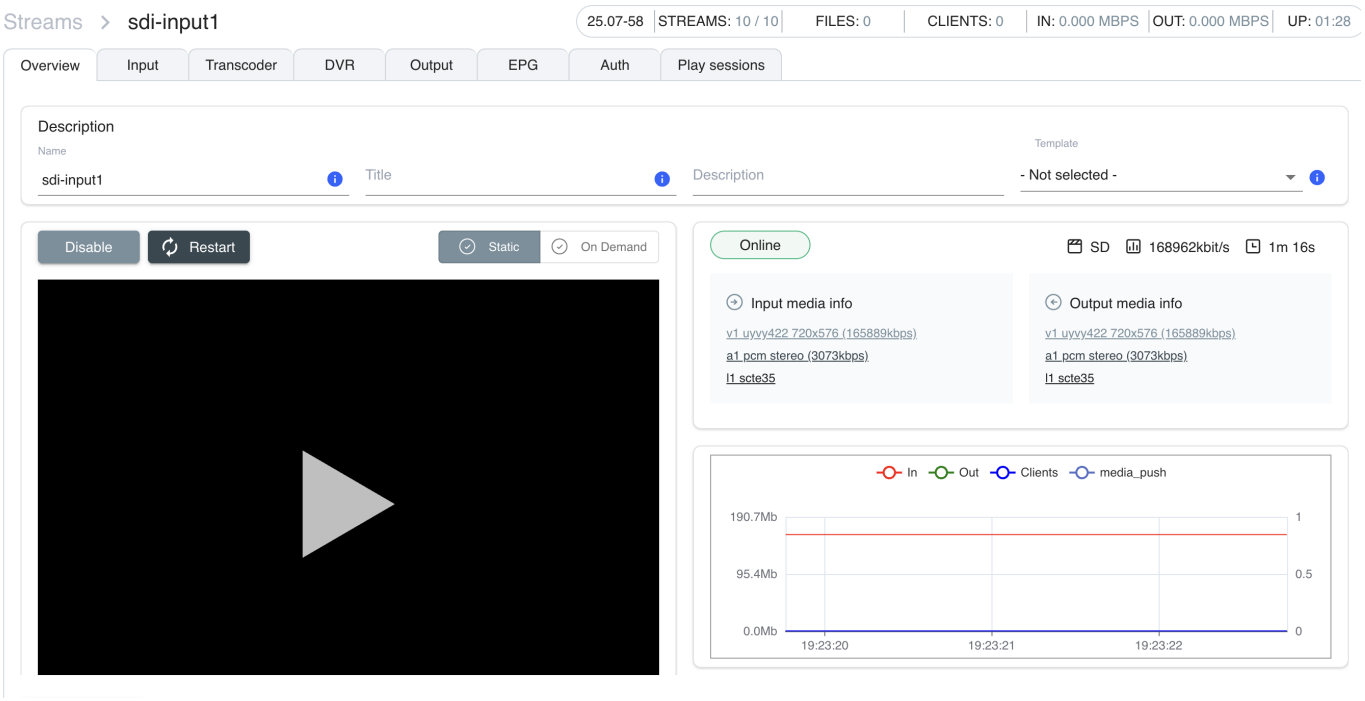
3.3.9 Conclusion

ASI Reader provides a reliable solution for receiving ASI signals in the mcaster system. The module provides flexible configuration, integration with various capture cards, and efficient flow transmission through multicast for further processing. Using multicast architecture allows scaling the system and ensuring fault tolerance.

3.4 SDI Coder

3.4.1 Overview

SDI Coder is a module within mcaster designed for capturing video and audio through SDI cards. The module provides high-quality reception of professional video signals and is one of the main stream input options in the mcaster system.



3.4.2 Supported Capture Cards

Dektec

- Full support for all Dektec models
- Automatic signal parameter detection
- Support for all SDI standards

Blackmagic Design

- Compatibility with Blackmagic cards
- Support for DeckLink series
- Integration with Blackmagic SDK

V4L Compatible Cards

- **Streamlabs** — streaming cards
- **Softlab** — professional capture cards
- **Magewell** — USB and PCI capture cards
- Any other cards with V4L2 support

3.4.3 Supported Formats

SDI-SD (Standard Definition)

- **Resolution**: 480i, 576i
- **Frame rates**: 25, 29.97, 30 fps
- **Features**: Analog teletext support (zvbi)
- **Application**: Archive materials, SD broadcasting

SDI-HD (High Definition)

- **Resolutions**: 720p, 1080i, 1080p
- **Frame rates**: 23.976, 24, 25, 29.97, 30, 50, 59.94, 60 fps
- **Standards**: HD-SDI, 3G-SDI

SDI-UHD (Ultra High Definition)

- **Resolutions**: 4K (2160p), 8K
- **Standards**: 6G-SDI, 12G-SDI
- **Application**: Modern UHD broadcasting

3.4.4 Input Signals

Video

- Main SDI video signal
- Automatic parameter detection
- Support for various color spaces

Multichannel Audio

- **Channels**: up to 16 audio channels
- **Formats**: PCM, AES/EBU
- **Sample rate**: 48 kHz (standard), 96 kHz (optional)
- **Bit depth**: 16, 20, 24 bit

Teletext and Subtitles

- **SDI-SD**: Analog teletext (zvbi)
- **SDI-HD**: Digital subtitles
- **Formats**: DVB, CEA-608, CEA-708
- **Encodings**: UTF-8, Latin-1

CEA-608/708 CLOSED CAPTIONS

Closed captions (CC) are text representation of the audio part of a TV program, movie, etc. It is a transcription or translation of the dialogue, sound effects, some relevant musical cues, and other relevant audio information in case when sound is unavailable or not clearly audible. Initially, closed captions were designed for deaf and hard of hearing people.

Mcaster is able to detect CEA-608/708 closed captions in SDI source streams and to read them. It is done automatically, so there's no need to configure it explicitly. The module reads the CEA-608/708 captions from an SDI stream, performs repackaging and then carries them within the MPEG-TS stream as an *H.264 SEI NALU*.

> **Note**
>
> *H.264 file consists of a number of* NAL Units*, i.e., Network Abstraction Layer Units,* SEI* refers to Supplemental Enhancement Information.

CEA-608 is a streaming, character-based format that allows for the transmission of up to 4 simultaneous channels of data. Mcaster adds 4 text tracks to those 4 channels (one for every channel). As a result, we have 4 text tracks and one video track carrying CEA-608/708 closed captions. You can later play those text tracks via WebVTT or TTML together with HLS, DASH, etc. video streams.

VBI TELETEXT (IMPORTANT FOR LEGACY BROADCASTING SYSTEMS)

VBI teletext is a critically important feature for legacy broadcasting systems that still transmit teletext and SD quality. **VBI** (Vertical Blanking Interval) is a gap in the sequence of lines which is used in analog television. During VBI no picture information is transmitted, but this area may contain such information as teletext or closed captions.

Mcaster allows reading EBU Teletext and subtitles (EBU Teletext subtitle data) from VBI in source streams received from an SDI card. Mcaster then retransmits them to MPTS or SPTS output.

When receiving video from an SDI card, Mcaster:

1. Decodes the received data
2. Reads the information about teletext from VBI
3. Compresses the data to be further transmitted over the Internet
4. Packages the stream with the teletext data into an MPEG-TS stream

VBI Teletext Configuration

To enable teletext reading, use the `ttxt_descriptors` parameter:

```
stream example_stream {
    input v4l2:// audio_device=plughw:1,0 ttxt_descriptors=0x100:rus:initial,0x888:rus:subtitle vbi_debug=true vbi_device=/dev/vbi0 video_device=/dev/video0;
}
```

For Decklink cards:

```
stream example_stream {
    input decklink://0 pixel=10 ttxt_descriptors=0x100:rus:initial,0x888:rus:subtitle;
}
```

For Decklink cards with NVENC transcoding:

```
stream example-stream {
  input decklink://2 pixel=10 ttxt_descriptors=0x100:rus:initial,0x888:rus:subtitle;
  transcoder deviceid=0 external=false hw=nvenc vb=5000k vcodec=h264 open_gop=false preset=veryfast size=3840x2160:fit:#000000 ab=128k split_channels=false;
}
```

Teletext Parameters

The `ttxt_descriptors` require the following parameters:

- `page` — page number according to the format `0x[teletext_magazine_number][teletext_page_number]`
- `lang` — the language of teletext according to the ISO 639-2 standard
- `type` — teletext page type:
- `initial` — initial teletext page
- `subtitle` — page with subtitles
- `impaired` — teletext subtitle page for hearing impaired people

Example: `ttxt_descriptors=0x100:rus:initial,0x888:rus:subtitle`

> **Note**
>
> This feature is especially important for integration with legacy broadcasting systems that continue to use analog teletext in SD quality.

3.4.5 Timestamp Correction Subsystem

Automatic Alignment

- **Timestamp jitter**: Automatic stabilization
- **Algorithms**: Adaptive filtering, median filtering
- **Accuracy**: ±1 µs

Lost Frame Correction

- **Detection**: Automatic identification of missing frames
- **Recovery**: Duplication of previous frame or interpolation
- **Logging**: Recording all corrections in log

Correction Settings

```
timestamp_correction:
  enabled: true
  jitter_threshold: 1000  # microseconds
  frame_drop_detection: true
  interpolation_method: "duplicate"  # duplicate, interpolate
  log_corrections: true
```

3.4.6 Module Configuration

Basic Settings

```
sdi_coder:
  device: "/dev/video0"
  input_format: "auto"  # auto, 1080i50, 720p60, etc.
  audio_channels: 8
  enable_teletext: true
  timestamp_correction: true
```

Capture Parameters

```
capture_settings:
  video_buffer_size: 10  # seconds
  audio_buffer_size: 5   # seconds
  drop_frames_on_overflow: true
  sync_mode: "hardware"  # hardware, software
```

Output Settings

```
output_settings:
  format: "mpegts"
  bitrate: "auto"  # auto or specific value
  gop_size: 30
  audio_codec: "aac"
  video_codec: "h264"
```

3.4.7 Integration with mcaster

As Stream Source

SDI Coder can be configured as a source for: - Live streaming - DVR recording - Transcoding - Network distribution

Stream Configuration Example

```
streams:
  sdi_main:
    source: "sdi_coder:///dev/video0"
    output: "rtmp://server/live/stream"
    transcoder:
      video:
        codec: "h264"
        bitrate: "5000k"
        resolution: "1920x1080"
      audio:
        codec: "aac"
        bitrate: "128k"
        channels: 2
```

3.4.8 Monitoring and Diagnostics

Capture Statistics

- Frame rate (current/average)
- Video and audio bitrate
- Number of timestamp corrections
- Synchronization status

Logging

```
logging:
  level: "info"  # debug, info, warning, error
  log_timestamp_corrections: true
  log_frame_drops: true
  log_device_status: true
```

Prometheus Metrics

- `sdi_coder_fps` — frame rate
- `sdi_coder_bitrate` — bitrate
- `sdi_coder_timestamp_corrections` — number of corrections
- `sdi_coder_frame_drops` — dropped frames

3.4.9 Troubleshooting

Common Issues

NO SIGNAL

1. Check SDI cable connection
2. Ensure correct device selection
3. Check input signal format

SYNCHRONIZATION ISSUES

1. Enable timestamp correction
2. Check SDI signal quality
3. Configure buffering parameters

AUDIO ISSUES

1. Check audio channel settings
2. Ensure audio format support
3. Check audio/video synchronization

Diagnostic Commands

```
# Check available devices
v4l2-ctl --list-devices

# Information about current signal
v4l2-ctl --device=/dev/video0 --all

# Capture test
ffmpeg -f v4l2 -i /dev/video0 -t 10 test.mp4
```

3.4.10 System Requirements

Hardware Requirements

- **CPU**: Minimum 4 cores for HD, 8 cores for UHD
- **RAM**: 8 GB for HD, 16 GB for UHD
- **Network**: Gigabit Ethernet
- **Storage**: SSD for buffering

Software Requirements

- Linux kernel 4.19+
- V4L2 support
- FFmpeg 4.0+
- Appropriate drivers for SDI card

3.4.11 Conclusion

SDI Coder provides a professional solution for capturing SDI signals in the mcaster system. The module ensures high reliability, automatic timestamp correction, and support for a wide range of equipment, making it an ideal choice for professional broadcasting.

3.5 HDMI Encoder

The HDMI encoder as part of Mcaster is designed to receive video signals from medium and entry-level consoles and prepare content for further transmission via various interfaces: SDI, NDI, ST2110 or already in compressed form.

3.5.1 Recommended Hardware

For HDMI capture, entry-level Blackmagic Decklink cards are recommended. They have the following characteristics:

- Affordable cost
- Sufficient reliability for short-term capture
- Ease of setup and use

> **Warning**
>
> For 24/7 operation, Blackmagic Decklink cards are not recommended due to possible issues with timecodes and stability during long-term use.

3.5.2 Drivers

All necessary drivers for working with the HDMI encoder are already included in the InfraMedia distribution and do not require additional installation.

3.5.3 HDMI Capture Setup

To check available capture devices, run the command:

```
BlackmagicFirmwareUpdater status
```

If successful, you will see a list of available capture devices.

Basic Stream Configuration

Configure the stream for HDMI capture as follows:

```
stream sdi {
  input decklink://0;
}
```

Mcaster will connect to the specified first device (0) and launch autoconfiguration to search for active resolution.

Manual Mode Configuration

Some Decklink models do not support automatic search for active resolution. For them, it is necessary to specify the mode manually using the `mode` and `vinput` options.

For example, for Intensity Pro with a connected HDMI source of 720p and 50 fps:

```
stream sdi {
  input decklink://0 mode=hp50 vinput=hdmi;
}
```

Web Interface Configuration

You can configure HDMI capture parameters through the Mcaster UI:

1. Go to the **Streams** tab on the **Media** page in the side menu
2. Open the settings of a stream configured for HDMI capture (with source `decklink://0`)
3. Go to the **Input** tab and click **Options**
4. Set the necessary parameter values in the **Decklink** section

3.5.4 HDMI Stream Transcoding

To transcode the captured HDMI stream, add the `transcoder` directive to the stream settings:

```
stream sdi {
  input decklink://0;
  transcoder vb=3096k ab=64k;
}
```

> **Note**
>
> The transcoding parameter `external=false` is used by default for HDMI and other "raw" video streams, preventing excessive server load.

Benefits of the New Approach

- Improved video quality by avoiding double transcoding
- Server resource savings
- "Seamless" switching between HDMI and other stream sources
- Ease of configuration through the web interface

> **Warning**
>
> If you do not specify transcoding settings in `transcoder`, the stream **will not** work.

3.5.5 Deinterlacing

To improve video quality, Mcaster can eliminate interlacing in progressive streams using the **CUDA yadif** deinterlacing method:

```
stream test {
  input decklink://1 vinput=sdi;
  transcoder vb=4000k hw=nvenc preset=slow fps=50 deinterlace=yadif ab=128k;
}
```

3.5.6 SD Video Capture

Mcaster supports video with non-square pixels (anamorphic video) when capturing from HDMI cards. This is especially important for SD (standard definition) quality.

To preserve proportions in the output video without image distortion, specify the **sar** of the input stream:

```
stream test {
  input decklink://1 vinput=hdmi sar=16:11;
}
```

Mcaster calculates the output video resolution based on `sar`. For example, with `sar=16:11`, incoming anamorphic video 720x576 will pass through Mcaster with 1048x576 resolution.

3.5.7 Duplex Mode Operation

Mcaster allows setting duplex mode for Decklink HDMI cards. In this mode, ports can be used individually for input or output, or as a combination of input and output.

For more information on setting up duplex mode, see Duplex Mode Operation.

3.5.8 Limitations and Recommendations

Time Limitations

Blackmagic Decklink cards have limitations during long-term use:

- High probability of timecode shifts
- Incorrect timecode transmission
- Instability during 24/7 operation

Recommendations for Critical Systems

For use in mission-critical systems where reliability and stability are important, it is recommended to use Dektec cards instead of Blackmagic Decklink.

Testing New Cards

If you want to test other capture cards for addition to the recommended list - contact us for testing.

3.6 SRT Reader

3.6.1 Overview

SRT Reader is a module within Mcaster that receives publications via the SRT (Secure Reliable Transport) protocol or captures streams from other servers via SRT. The module provides reliable video stream transmission with minimal latency and automatic recovery of lost packets.

3.6.2 Operating Principles

Receiving Publications

The module can receive SRT streams from external sources:

- **Publication** — receiving streams from encoders or other servers
- **Capture** — connecting to remote SRT servers
- **SPTS Support** — working with Single Program Transport Stream

SRT Protocol

- **Reliability** — automatic recovery of lost packets
- **Security** — built-in encryption using passphrase
- **Low latency** — optimized for live video
- **Adaptability** — automatic adjustment to network quality

3.6.3 Configuration

Basic Setup for Receiving Publications

```
stream input-srt {
  input publish://;
  srt_publish {
    port 5912;
    latency 40;
  }
}
```

Configuration Parameters

Parameter	Description	Required	Example
`input publish://`	Allows publication to stream	Yes	`publish://`
`port`	Port for receiving SRT publication	Yes	`5912`
`latency`	Latency in milliseconds	No	`40`
`passphrase`	Encryption key	No	`mysecretkey`

Advanced Configuration

```
stream secure-srt {
  input publish://;
  srt_publish {
    port 5913;
    latency 60;
    passphrase "mysecretkey123";
  }

  # Additional settings
  buffer_size 8192;
  timeout 5000;
}
```

Stream Capture Configuration

```
stream capture-srt {
  input srt://remote-server:5912?passphrase=mysecretkey;
  output rtmp://server/live/captured;
}
```

3.6.4 SRT Options

passphrase

- **Purpose**: Encryption key for stream protection
- **Requirement**: Must be identical on both ends of connection
- **Format**: String of arbitrary length
- **Recommendation**: Use complex keys for security

latency

- **Purpose**: Buffer latency configuration
- **Behavior**: Affects stability but not critical for operation
- **Values**: Usually 20-200 milliseconds
- **Default**: 40 milliseconds

3.6.5 Publication Testing

Sending Stream via FFmpeg

```
# Publishing local file
ffmpeg -re -i input.mp4 -c copy -f mpegts srt://localhost:5912

# Publishing from camera
ffmpeg -f v4l2 -i /dev/video0 -c:v libx264 -preset ultrafast -tune zerolatency -f mpegts srt://localhost:5912

# Publishing with passphrase
ffmpeg -re -i input.mp4 -c copy -f mpegts "srt://localhost:5912?passphrase=mysecretkey"
```

Sending Stream from Another Server

```
# From another mcaster
ffmpeg -re -i input.mp4 -c copy -f mpegts srt://mcaster-server:5912

# From OBS Studio
# Configure SRT Output in OBS with address mcaster-server:5912
```

3.6.6 Monitoring

SRT-Specific Parameters

ROUND TRIP TIME (RTT)

```
{
  "stats": {
    "input": {
      "srt": {
        "rtt": 25.5  // Round Trip Time in milliseconds
      }
    }
```

```
  }
}
```

- **Description**: Total delay for feedback
- **Normal value**: 10-50 ms
- **Problematic value**: >100 ms
- **Action**: Check network quality when RTT is high

REAL LATENCY

```
{
  "stats": {
    "input": {
      "srt": {
        "latency": 35.2  // Real latency in milliseconds
      }
    }
  }
}
```

- **Description**: Actual latency on receiving side
- **Variability**: Changes due to packet losses
- **Normal value**: 20-80 ms
- **Monitoring**: Track value stability

RETRANSMITTED PACKETS

```
{
  "stats": {
    "input": {
      "srt": {
        "retransmitted_packets": 15  // Number of retransmitted packets
      }
    }
  }
}
```

- **Description**: Number of packets sent again
- **Normal value**: 0-50 packets per minute
- **Problematic value**: >100 packets per minute
- **Cause**: Poor network quality

General MPEGTS Reader Metrics

```
{
  "stats": {
    "input": {
      "packets_received": 125000,
      "packets_lost": 5,
      "bitrate": 5000000,
      "fps": 25.0
    }
  }
}
```

3.6.7 Usage Examples

Simple Publication

```
stream live_channel {
  input publish://;
  srt_publish {
    port 5912;
    latency 40;
  }
  output rtmp://server/live/stream;
}
```

Secure Publication

```
stream secure_channel {
  input publish://;
  srt_publish {
    port 5913;
    latency 60;
    passphrase "complex_secret_key_2024";
  }
  output rtmp://server/live/secure;
}
```

Multiple Streams

```
# Stream 1
stream channel_1 {
  input publish://;
  srt_publish {
    port 5912;
    latency 40;
  }
  output rtmp://server/live/ch1;
}

# Stream 2
stream channel_2 {
  input publish://;
  srt_publish {
    port 5913;
    latency 40;
  }
  output rtmp://server/live/ch2;
}
```

Transcoder Integration

```
stream transcoded_srt {
  input publish://;
  srt_publish {
    port 5914;
    latency 50;
  }

  transcoder {
    video {
      codec h264;
      bitrate 5000k;
    }
    audio {
      codec aac;
      bitrate 128k;
    }
  }

  output rtmp://server/live/transcoded;
}
```

3.6.8 Troubleshooting

Connection Issues

CANNOT CONNECT TO PORT

1. **Check port availability** — ensure port is not occupied
2. **Check firewall** — allow incoming connections
3. **Check configuration** — ensure settings are correct
4. **Check module logs** for errors

HIGH RTT

1. **Check network quality** between client and server
2. **Increase latency** for stabilization
3. **Check server load**
4. **Consider using CDN**

FREQUENT PACKET RETRANSMISSIONS

1. **Check network stability**
2. **Reduce stream bitrate**
3. **Check encoder settings**
4. **Monitor internet connection quality**

Diagnostic Commands

```
# Check port availability
netstat -tuln | grep 5912

# Connection test
telnet localhost 5912

# Network traffic monitoring
tcpdump -i any -n port 5912

# Check SRT metrics
curl -X GET "http://localhost:8080/api/stream_get?name=input-srt"
```

3.6.9 Configuration Recommendations

Optimal Latency Values

- **Stable network**: 20-40 ms
- **Unstable network**: 60-120 ms
- **Satellite connection**: 200-500 ms
- **Mobile network**: 100-200 ms

Security

- **Use complex passphrases** — minimum 16 characters
- **Regularly change keys** — every 30-90 days
- **Restrict port access** — use firewall
- **Monitor connections** — track suspicious activity

Performance

- **Sufficient bandwidth** — minimum 2x stream bitrate
- **Stable internet connection** — to minimize losses
- **Encoder optimization** — configure for SRT
- **Resource monitoring** — CPU, memory, network

Critical Parameter Monitoring

- `stats.input.srt.rtt` — network quality
- `stats.input.srt.retransmitted_packets` — connection stability
- `stats.input.srt.latency` — real latency

3.6.10 Conclusion

SRT Reader provides reliable and secure video stream transmission with minimal latency. The module supports both receiving publications and capturing streams, making it a universal solution for various broadcasting scenarios. Built-in monitoring and diagnostics allow efficient management of transmission quality and quick resolution of emerging issues.

3.7 DVB Reader

3.7.1 Overview

DVB Reader is a module within mcaster that allows receiving video directly from DVB capture cards. The module supports various DVB standards (DVB-S/S2, DVB-T/T2, DVB-C) and provides reliable reception of FTA (Free-to-Air) channels or descrambled content.

3.7.2 Supported Standards

DVB-S/S2 (Satellite Broadcasting)

- **Frequency range**: 950-2150 MHz
- **Modulations**: QPSK, 8PSK, 16APSK, 32APSK
- **Polarization**: Horizontal (H) and Vertical (V)
- **Symbol rates**: 1000-45000 KS/s

DVB-T/T2 (Terrestrial Broadcasting)

- **Frequency range**: 174-862 MHz
- **Modulations**: QPSK, 16QAM, 64QAM, 256QAM
- **Bandwidths**: 6, 7, 8 MHz
- **Modes**: 2K, 4K, 8K

DVB-C (Cable Broadcasting)

- **Frequency range**: 47-862 MHz
- **Modulations**: 16QAM, 32QAM, 64QAM, 128QAM, 256QAM
- **Symbol rates**: 1000-7000 KS/s

3.7.3 DVB Card Configuration

Basic Configuration Structure

```
dvb_card card_name {
    system standard;
    adapter adapter_number;
    frontend frontend_number;
    frequency frequency;
    symbol_rate symbol_rate;
    polarization polarization;
    modulation modulation;
    bandwidth bandwidth;
    plp_stream_id stream_id;
    disabled;
    comment "description";
}
```

Configuration Parameters

Parameter	Description	Required	Example
`system`	DVB standard	Yes	`dvbs2`, `dvbt`, `dvbc`
`adapter`	Adapter number	Yes	`0`, `1`, `2`
`frontend`	Frontend number	Yes	`0`, `1`, `2`, `3`
`frequency`	Frequency in Hz	Yes	`195028615`
`symbol_rate`	Symbol rate	Yes	`29500`
`polarization`	Polarization (for DVB-S)	No	`h`, `v`
`modulation`	Modulation type	No	`qam256`, `qpsk`
`bandwidth`	Bandwidth	No	`5000000`
`plp_stream_id`	PLP stream ID	No	`4`
`disabled`	Disable card	No	-
`comment`	Comment	No	`"13E high vertical"`

DVB-S2 Configuration Example

```
dvb_card a0 {
    system dvbs2;
    adapter 1;
    frontend 3;
    frequency 195028615;
    symbol_rate 29500;
    polarization v;
    modulation qam256;
    bandwidth 5000000;
    plp_stream_id 4;
    comment "13E high vertical";
}
```

DVB-T2 Configuration Example

```
dvb_card terrestrial {
    system dvbt2;
    adapter 0;
    frontend 0;
    frequency 474000000;
    bandwidth 8000000;
    modulation qam256;
    comment "DVB-T2 multiplex";
}
```

DVB-C Configuration Example

```
dvb_card cable {
    system dvbc;
    adapter 0;
    frontend 0;
    frequency 474000000;
    symbol_rate 6875;
    modulation qam256;
    comment "Cable network";
}
```

3.7.4 Stream Configuration

Basic Stream

```
stream ort {
  input mpts-dvb://a0?program=15;
}
```

Stream Parameters

Parameter	Description	Required	Example
`mpts-dvb://`	DVB protocol	Yes	`mpts-dvb://`
`card_name`	Configured card name	Yes	`a0`, `terrestrial`
`program`	Program number	Yes	`15`, `1`, `2`

Advanced Stream Configuration

```
stream hd_channel {
  input mpts-dvb://a0?program=15;

  transcoder {
    video {
      codec h264;
      bitrate 5000k;
    }
    audio {
      codec aac;
      bitrate 128k;
    }
  }
}
```

Multiple Streams

```
# Main channel
stream main_channel {
  input mpts-dvb://a0?program=15;
}

# Secondary channel
stream secondary_channel {
  input mpts-dvb://a0?program=16;
}
```

3.7.5 Supported Capture Cards

DVB-S/S2 Cards

- **TBS** — satellite reception card series
- **DekTec** — professional capture cards
- **Hauppauge** — home use cards
- **TechnoTrend** — budget solutions

DVB-T/T2 Cards

- **Hauppauge** — terrestrial reception cards
- **PCTV** — USB tuners
- **AverMedia** — digital TV cards

DVB-C Cards

- **Hauppauge** — cable TV cards
- **PCTV** — cable USB tuners
- **TechnoTrend** — cable network cards

3.7.6 Troubleshooting

Signal Issues

NO SIGNAL

1. **Check connection** of antenna/cable
2. **Ensure correctness** of frequency and parameters
3. **Check polarization** (for DVB-S)
4. **Check capture card drivers**

WEAK SIGNAL

1. **Check quality** of antenna and cables
2. **Ensure correctness** of antenna direction
3. **Check interference** from other devices
4. **Consider signal amplifier**

POOR QUALITY

1. **Check modulation settings**
2. **Ensure correctness** of symbol rate
3. **Check interference** and reflections
4. **Optimize antenna placement**

Capture Card Issues

CARD NOT DETECTED

1. **Check card connection**
2. **Ensure compatibility** with system
3. **Check drivers** and firmware
4. **Check device access permissions**

DRIVER ERRORS

1. **Update card drivers**
2. **Check compatibility** with Linux kernel
3. **Reboot system**
4. **Check conflicts** with other devices

Diagnostic Commands

```
# Check available DVB devices
ls /dev/dvb/

# Card information
dvbv5-scan -a 0 -f 195028615 -s 29500 -p v -m qam256

# Signal check
dvbv5-zap -a 0 -f 195028615 -s 29500 -p v -m qam256

# Statistics monitoring
cat /proc/dvb/adapter0/frontend0/statistics
```

3.7.7 Configuration Recommendations

Reception Optimization

DVB-S/S2

- **Use quality antenna** of appropriate size
- **Properly configure polarization** and frequency
- **Check symbol rate** of transponder
- **Use quality cables** with minimal losses

DVB-T/T2

- **Check coverage** in your region
- **Use directional antenna** for better reception
- **Properly configure bandwidth**
- **Check multiplex modulation**

DVB-C

- **Ensure compatibility** with cable network
- **Check symbol rate** of provider
- **Configure correct modulation**
- **Check quality** of cable connection

Security and Stability

- **Regularly update drivers** of capture cards
- **Monitor signal quality** in real-time
- **Use backup cards** for critical channels
- **Maintain logs** for problem diagnosis

Performance

- **Optimize settings** for specific content
- **Use hardware acceleration** when possible
- **Monitor system load** with multiple streams
- **Plan redundancy** for important channels

3.7.8 Conclusion

DVB Reader provides reliable and efficient video reception from DVB capture cards. Support for various DVB standards, flexible configuration, and built-in monitoring make the module an ideal solution for professional broadcasting. Proper parameter configuration and regular signal quality monitoring ensure stable operation in complex reception conditions.

3.8 DVB-WebVTT

3.8.1 Overview

DVB-WebVTT is a module in mcaster that receives subtitles from DVB in graphical format, recognizes them using OCR (Optical Character Recognition), and converts them to textual WebVTT subtitles. The resulting subtitles are suitable for playback on modern devices, including phones and televisions.

3.8.2 Operating Principles

DVB Subtitle Processing

1. **DVB subtitle reception** — the module receives subtitles in graphical format from MPEG-TS flow
2. **OCR recognition** — automatic text recognition from subtitle images
3. **WebVTT conversion** — transformation into textual WebVTT format
4. **Language separation** — automatic separation of subtitles by languages
5. **Manifest insertion** — adding subtitles to HLS and DASH manifests

Supported Formats

- **Input**: DVB subtitles in graphical format
- **Output**: WebVTT textual subtitles
- **Containers**: HLS, DASH
- **Languages**: Automatic detection and separation by languages

3.8.3 Configuration

Basic Setup

```
stream vtt1 {
  input udp://239.0.0.1:1234;
  dvbocr replace;
}
```

Configuration Parameters

Parameter	Description	Required	Example
`input`	Input flow with DVB subtitles	Yes	`udp://239.0.0.1:1234`
`dvbocr`	Subtitle processing mode	Yes	`replace`

Dvbocr Operation Modes

`replace`

- **Action**: Replaces graphical DVB subtitles with textual WebVTT
- **Result**: Output flow contains only textual subtitles
- **Application**: Standard mode for most cases

`add`

- **Action**: Adds textual subtitles to existing graphical ones
- **Result**: Output flow contains both types of subtitles
- **Application**: For compatibility with old devices

Extended Configuration

```
stream vtt1 {
  input udp://239.0.0.1:1234;
  dvbocr replace;

  # Additional parameters
  subtitle_languages auto;  # Automatic language detection
  ocr_confidence 0.8;      # Minimum recognition confidence
  subtitle_delay 0;        # Subtitle delay in seconds
}
```

3.8.4 OCR Technology

Recognition Algorithms

- **Neural networks** — modern machine learning algorithms
- **Adaptive processing** — adjustment to image quality
- **Multilingual recognition** — support for various languages
- **Context analysis** — improving recognition accuracy

Quality Optimization

- **Image preprocessing** — improving contrast and clarity
- **Noise filtering** — removing compression artifacts
- **Result validation** — checking recognized text correctness
- **Error correction** — fixing typical OCR errors

3.8.5 Output Formats

WebVTT Subtitles

```
WEBVTT

00:00:01.000 --> 00:00:04.000
This is an example of textual subtitles

00:00:05.000 --> 00:00:08.000
Recognized from DVB flow
```

HLS Manifest

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXTINF:10.0,
segment_001.ts
#EXTINF:10.0,
segment_002.ts

# Subtitles for different languages
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",LANGUAGE="ru",NAME="Russian",DEFAULT=YES,URI="subtitles_ru.vtt"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",LANGUAGE="en",NAME="English",DEFAULT=NO,URI="subtitles_en.vtt"
```

DASH Manifest

```
<AdaptationSet mimeType="text/vtt" lang="ru">
  <SegmentTemplate media="subtitles_ru_$Number$.vtt" initialization="subtitles_ru_init.vtt"/>
</AdaptationSet>
<AdaptationSet mimeType="text/vtt" lang="en">
  <SegmentTemplate media="subtitles_en_$Number$.vtt" initialization="subtitles_en_init.vtt"/>
</AdaptationSet>
```

3.8.6 Automatic Language Separation

Language Detection

The module automatically determines subtitle language based on **language codes** in DVB flow

3.8.7 Troubleshooting

Recognition Problems

LOW RECOGNITION QUALITY

1. **Check quality** of incoming DVB signal
2. **Check language support** by support, possibly it is not supported

NO SUBTITLES IN OUTPUT

1. **Check presence** of DVB subtitles in input flow
2. **Ensure correctness** of dvbocr configuration
3. **Check module logs** for errors

Performance

- **Sufficient CPU resources** — for real-time OCR processing
- **Memory optimization** — for subtitle frame buffering
- **Efficient disk system** — for output file writing

3.8.8 Device Compatibility

Supported Devices

- **Modern televisions** — with HLS/DASH support
- **Smartphones and tablets** — iOS, Android
- **Web browsers** — Chrome, Firefox, Safari, Edge
- **Streaming devices** — Apple TV, Chromecast, Fire TV

Subtitle Formats

- **WebVTT** — standard for HTML5 video
- **HLS subtitles** — embedded in m3u8 manifests
- **DASH subtitles** — separate adaptation sets

3.8.9 Conclusion

DVB-WebVTT module provides a modern solution for converting graphical DVB subtitles to textual WebVTT subtitles. Automatic recognition, multilingual support, and integration with HLS/DASH make it an indispensable tool for ensuring content accessibility on modern devices.

3.9 DVR

Mcaster DVR module writes video archives to disk and enables playback of recorded content. This functionality allows users to pause live broadcasts, rewind to earlier parts of programs, and access archived content from previous days.

3.9.1 Overview

The DVR module provides comprehensive video recording and archive management capabilities. It supports various playback scenarios including:

- Pausing live broadcasts to take breaks
- Watching current programs from the beginning
- Accessing yesterday's programs
- Bookmarking favorite programs for later viewing

Key Features

- **Continuous Recording**: Automatic archive creation with configurable retention periods
- **Multiple Playback Methods**: Support for different timeshift approaches
- **Middleware Integration**: Seamless integration with IPTV middleware systems
- **Flexible Storage**: Configurable storage locations and retention policies
- **Timeshift Support**: Both relative and absolute timeshift capabilities

3.9.2 Basic Configuration

Simple DVR Setup

```
stream example_channel {
  input udp://239.0.0.1:1234;
  dvr /storage 7d;
}
```

Where: * `/storage` - path to storage directory (stream name will be automatically added) * `7d` - retention period (7 days)

Advanced DVR Configuration

```
stream premium_channel {
  input udp://239.0.0.1:1234;
  dvr /storage 30d;
}
```

3.9.3 Playback Methods

EPG-Based Archive Playback

For middleware systems with precise EPG schedules, use EPG-based archive URLs:

```
http://MCASTER-IP/STREAM_NAME/archive-START_TIME-DURATION.m3u8?event=true
```

Where: * `START_TIME` - program start time in UTC (Epoch time) * `DURATION` - program duration in seconds

Example:

```
http://192.168.1.100/news_channel/archive-1717677139-2116.m3u8?event=true
```

This method allows:

- Seeking within the program
- Pausing and resuming
- Fast forward playback

Event Playlists for Live Content

For current broadcasts, use event playlists that automatically switch from EVENT to VOD mode:

```
http://MCASTER-IP/STREAM_NAME/archive-START_TIME-DURATION.m3u8?event=true
```

Important Notes:

- Event playlists don't support rewinding in native Safari
- Requires custom JavaScript timeline implementation for TV applications
- Automatic switching between EVENT and VOD modes

3.9.4 Timeshift Capabilities

Relative Timeshift

Create delayed streams with fixed time offsets:

```
stream delayed_channel {
  input timeshift://original_channel/3600;
}
```

This creates a stream with 1-hour delay (3600 seconds).

Absolute Timeshift

For personalized timeshift access, use absolute timeshift URLs:

HLS Playback:

```
http://MCASTER-IP/STREAM_NAME/timeshift_abs-TIMESTAMP.m3u8
```

HTTP MPEG-TS Playback:

```
http://MCASTER-IP/STREAM_NAME/timeshift_abs-TIMESTAMP.ts
```

Where `TIMESTAMP` is the absolute time in UTC (Epoch time).

Handling Archive Gaps

When archive contains gaps (e.g., source downtime), use the `ignore_gaps=true` parameter:

```
http://MCASTER-IP/STREAM_NAME/timeshift_abs-123123123.m3u8?ignore_gaps=true
```

This allows playback to continue by skipping gaps in the archive.

3.9.5 Middleware Integration

EPG-VOD Method

For playing archived content, use EPG-based URLs:

```
http://MCASTER-IP/STREAM_NAME/archive-START_TIME-DURATION.m3u8?event=true
```

Implementation Requirements:

- Precise EPG schedule maintenance
- UTC time handling (avoid local time)
- Viewing time storage in middleware database
- Automatic program switching at playback end

Event Playlist Method

For current broadcasts, use event playlists with automatic mode switching:

```
http://MCASTER-IP/STREAM_NAME/archive-START_TIME-DURATION.m3u8?event=true
```

Key Features:

- Automatic EVENT to VOD switching
- Pause and resume functionality
- Custom "live" button implementation required
- EPG-based program continuation

3.9.6 Scalability Considerations

Timeshift_abs Limitations

The `timeshift_abs` method has significant scalability limitations:

- **Session Management**: Flussonic uses probabilistic session joining based on client IP, channel name, protocol, and token
- **Consecutive Requests**: Multiple timeshift_abs requests may be treated as the same session
- **Viewing Distortion**: Can cause playback issues with multiple simultaneous users
- **Token Requirements**: New tokens should be passed for each timeshift_abs request

Scaling Solutions

For high-user scenarios, consider these approaches:

Proportional DVR Servers:

- Deploy DVR servers proportionally to the number of viewers
- Each server handles a subset of users
- Reduces session conflicts

Alternative Methods:

- Use relative timeshift for popular content
- Implement middleware-based timeshift management
- Consider CDN integration for archive distribution

3.9.7 Storage Management

Retention Policies

Configure retention periods based on content type and storage capacity:

```
stream news_channel {
  input udp://239.0.0.1:1234;
  dvr /storage 7d;  # 7 days retention
}
```

```
stream movies_channel {
  input udp://239.0.0.1:1234;
  dvr /storage 30d;  # 30 days retention
}
```

Storage Optimization

- **Segment Duration**: Balance between seek granularity and storage efficiency
- **Compression**: Use appropriate video compression for archive storage
- **Cleanup**: Implement automatic cleanup of expired archives
- **Monitoring**: Track storage usage and archive health

3.9.8 Performance Monitoring

Key Metrics

- **Recording Performance**: Monitor archive creation speed and reliability
- **Playback Performance**: Track timeshift request success rates
- **Storage Utilization**: Monitor disk usage and cleanup efficiency
- **Session Management**: Track timeshift_abs session conflicts

Health Checks

- **Archive Integrity**: Verify recorded content quality
- **Storage Availability**: Monitor disk space and I/O performance
- **Network Performance**: Track archive delivery metrics
- **Error Rates**: Monitor recording and playback failures

3.9.9 Troubleshooting

Common Issues

- **Archive Gaps**: Use `ignore_gaps=true` parameter or investigate source issues
- **Session Conflicts**: Implement proper token management for timeshift_abs
- **Storage Full**: Monitor retention policies and cleanup processes
- **Playback Errors**: Check archive integrity and network connectivity

Debug Tools

- **Archive Inspection**: Verify recorded content and timestamps
- **Session Logs**: Monitor timeshift session management
- **Storage Analysis**: Check disk usage and I/O patterns
- **Network Diagnostics**: Verify archive delivery performance

3.10 RTMP Reader

This module allows receiving video via RTMP protocol from broadcast software such as OBS or vMix.

3.10.1 Global Port Setup

To configure it, you need to enable the global rtmp port:

```
rtmp 1935;
```

Don't forget to unblock it on the firewall.

3.10.2 Stream Configuration

Then allow publishing in the stream:

```
stream s {
  input publish://;
}
```

3.10.3 Access Control

To restrict publishing capability, you need to specify a password:

```
stream s {
  input publish;
  password secretkey;
}
```

In this case, publishing should go to a stream with the name:

```
s?password=secretkey
```

3.10.4 RTMP Server

The RTMP server name will be:

```
rtmp://server-hostname/static
```

3.10.5 Supported Broadcast Software

The module works with the following broadcast software:

- OBS Studio
- vMix
- Wirecast
- XSplit Broadcaster
- Other RTMP-compatible applications

3.10.6 Security

For security, it is recommended to:

- Use passwords for all publications
- Configure firewall to restrict access to RTMP port
- Regularly change passwords
- Monitor active connections

3.10.7 Monitoring

This module does not have separate telemetry, but you can track:

- Number of active RTMP connections
- Publication status
- Connection errors
- Server resource usage

3.11 LiveStreamInput (LSI)

3.11.1 Overview

LiveStreamInput (LSI) is a module within mcaster that implements signal source management and automatic switching between primary and backup sources. The module ensures high broadcasting reliability through automatic switching when problems occur with the primary source.

3.11.2 Operating Principles

Automatic Switching

LSI automatically switches between sources under the following conditions: - **Missing frames** on the primary source - **Signal quality issues** - **Technical equipment failures**

Backup Source Verification

During primary source operation, LSI regularly performs test connections to secondary sources to ensure backup readiness.

Source Compatibility

LSI checks compatibility between primary and backup sources by: - Codecs - Audio tracks - Stream parameters - Other characteristics for seamless switching

3.11.3 Configuration

Basic Setup

```
stream Reg_1010_01_Kanal_ENC {
  input copy://Reg_1010_01_Kanal_sdi1;
  input copy://Reg_1010_01_Kanal_sdi2;
  title "01 FIRST CHANNEL";
  source_timeout 1;
}
```

Configuration Parameters

Parameter	Description	Required	Example
`input`	Video signal source	Yes	`copy://Reg_1010_01_Kanal_sdi1`
`title`	Stream name	No	`"01 FIRST CHANNEL"`
`source_timeout`	Switch timeout (seconds)	No	`1`

Advanced Configuration

```
stream Main_Channel {
  input copy://primary_source source_timeout=10;
  input copy://backup_source_1;
  input copy://backup_source_2;
  title "Main Channel";
  source_timeout 2;
}
```

3.11.4 Monitoring via HTTP API

Main API Methods

- `streams_list` — list of all streams
- `stream_get` — detailed stream information

Key Stats Parameters

RECONNECTIONS AND SWITCHES

```
{
  "stats": {
    "input": {
      "retries": 15,           // Number of reconnections
      "input_switches": 3,     // Number of switches between sources
      "num_sec_on_primary_input": 86400,    // Time working on primary source
      "num_sec_on_secondary_input": 3600    // Time working on backup source
    }
  }
}
```

BACKUP SOURCE STATUS

```
{
  "stats": {
    "input": {
      "valid_secondary_inputs": 2,      // Number of working backup sources
      "invalid_secondary_inputs": 1,    // Number of failed backup sources
      "divergent_inputs": 0             // Number of incompatible sources
    }
  }
}
```

3.11.5 Monitoring Parameters Interpretation

Reconnections (`stats.input.retries`)

- **Normal value**: 0-5 reconnections per day
- **Problematic value**: >10 reconnections per day
- **Action**: When there are many reconnections, check source stability

Switches (`stats.input.input_switches`)

- **Normal value**: 0-2 switches per day
- **Problematic value**: >5 switches per day
- **Action**: Frequent switches indicate problems with the primary source

Time Working on Sources

Comparing `num_sec_on_primary_input` and `num_sec_on_secondary_input` shows: - **Backup efficiency** — how much time the system worked on backup - **Primary source quality** — frequency of backup usage - **Potential downtime** without automatic switching

Backup Source Status

WORKING BACKUPS (`stats.input.valid_secondary_inputs`)

- **Normal value**: >0 (working backup available)
- **Critical value**: 0 (no working backups)
- **Action**: When value is 0, immediate repair of backup sources is required

FAILED BACKUPS (`stats.input.invalid_secondary_inputs`)

- **Normal value**: 0-1
- **Problematic value**: >2
- **Action**: Consider removing or reducing payment for unstable sources

INCOMPATIBLE SOURCES (`stats.input.divergent_inputs`)

- **Normal value**: 0
- **Problematic value**: >0

- **Risk**: During failure, playback issues up to TV freezing are possible
- **Action**: Bring sources to unified format

3.11.6 Monitoring Examples

Checking Stream Status

```
# Getting list of streams
curl -X GET "http://localhost:8080/api/streams_list"

# Getting detailed stream information
curl -X GET "http://localhost:8080/api/stream_get?name=Reg_1010_01_Kanal_ENC"
```

Monitoring Script

```
#!/bin/bash

# Checking number of reconnections
retries=$(curl -s "http://localhost:8080/api/stream_get?name=Reg_1010_01_Kanal_ENC" | jq '.stats.input.retries')

if [ $retries -gt 10 ]; then
    echo "WARNING: High number of retries: $retries"
fi

# Checking availability of backup sources
valid_backups=$(curl -s "http://localhost:8080/api/stream_get?name=Reg_1010_01_Kanal_ENC" | jq '.stats.input.valid_secondary_inputs')

if [ $valid_backups -eq 0 ]; then
    echo "CRITICAL: No valid backup sources available"
fi
```

3.11.7 Troubleshooting

Frequent Reconnections

1. **Check stability** of primary source
2. **Increase source_timeout** to reduce sensitivity
3. **Check network settings** between mcaster and source
4. **Monitor logs** of LSI module

Missing Backup Sources

1. **Check availability** of backup sources
2. **Ensure correctness** of configuration
3. **Check network connectivity**
4. **Restore backup sources**

Source Incompatibility

1. **Bring sources** to unified format
2. **Check codecs** and parameters
3. **Ensure matching** audio tracks
4. **Configure unified** stream parameters

3.11.8 Configuration Recommendations

Optimal source_timeout Values

- **Stable sources**: 1-2 seconds
- **Unstable sources**: 3-5 seconds

- **Satellite sources**: 5-10 seconds

Number of Backup Sources

- **Minimum**: 1 backup source
- **Recommended**: 2-3 backup sources
- **Maximum**: 5 backup sources (to avoid complexity)

Monitoring and Alerts

```
alerts:
  - condition: "stats.input.valid_secondary_inputs == 0"
    severity: "critical"
    message: "No backup sources available"

  - condition: "stats.input.retries > 10"
    severity: "warning"
    message: "High number of reconnections"

  - condition: "stats.input.divergent_inputs > 0"
    severity: "warning"
    message: "Incompatible sources detected"
```

3.11.9 Conclusion

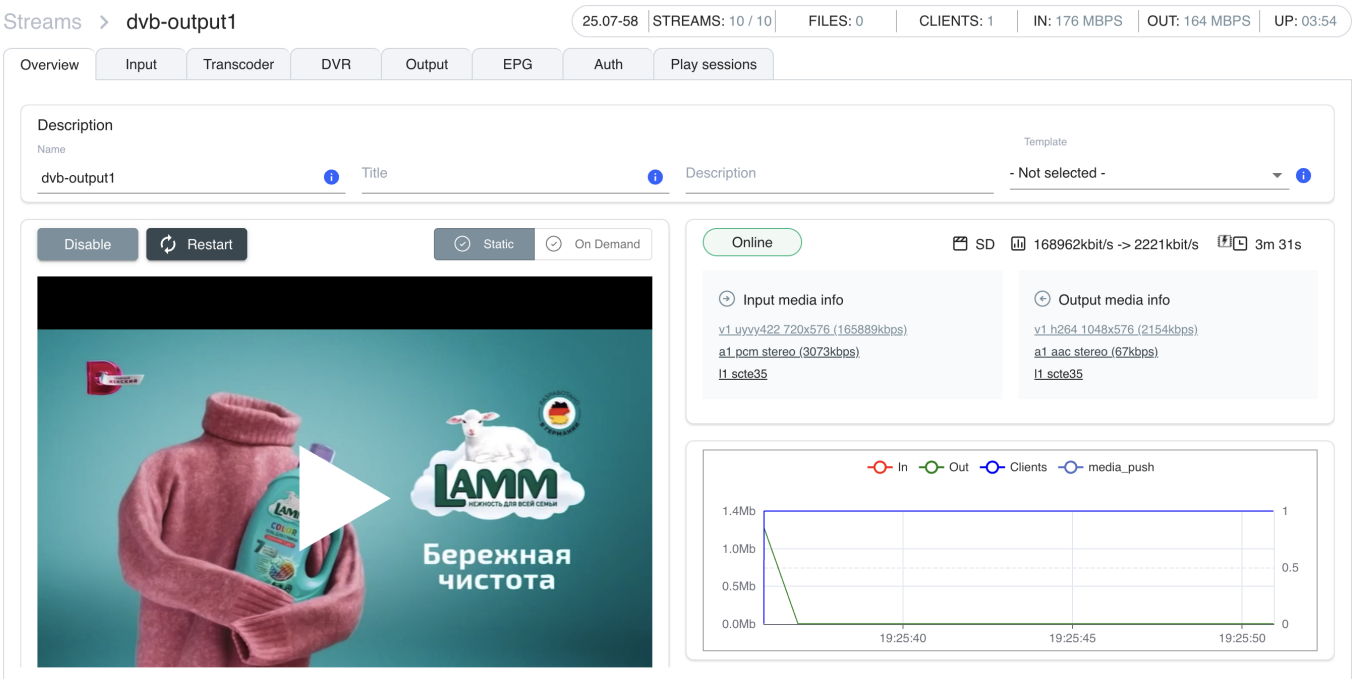
LiveStreamInput (LSI) ensures high broadcasting reliability through automatic switching between sources. Proper configuration and monitoring of the module allows minimizing service downtime and ensuring stable broadcasting even when problems occur with the primary signal source.

3.12 Transcoder

3.12.1 Transcoder

Overview

Transcoder is a module within mcaster that provides video stream transcoding for various broadcasting scenarios. The module supports both DVB-compatible CBR (Constant Bit Rate) quality generation and multibitrate (MBR) encoding for OTT (Over-The-Top) services.



Module Application

MAIN USAGE SCENARIOS

Connecting SDI and Compressed Video

- **Problem**: Need to transmit SDI signal in compressed format
- **Solution**: Transcoding SDI to H.264/H.265 for digital broadcasting
- **Result**: Compatibility with modern delivery systems

Bitrate Reduction

- **Problem**: High bitrate of incoming stream
- **Solution**: Transcoding with bitrate optimization
- **Result**: Network bandwidth savings

DVB and OTT Integration

- **Problem**: Different quality requirements for DVB and internet broadcasting
- **Solution**: Creating separate streams with different settings
- **Result**: Optimal quality for each delivery type

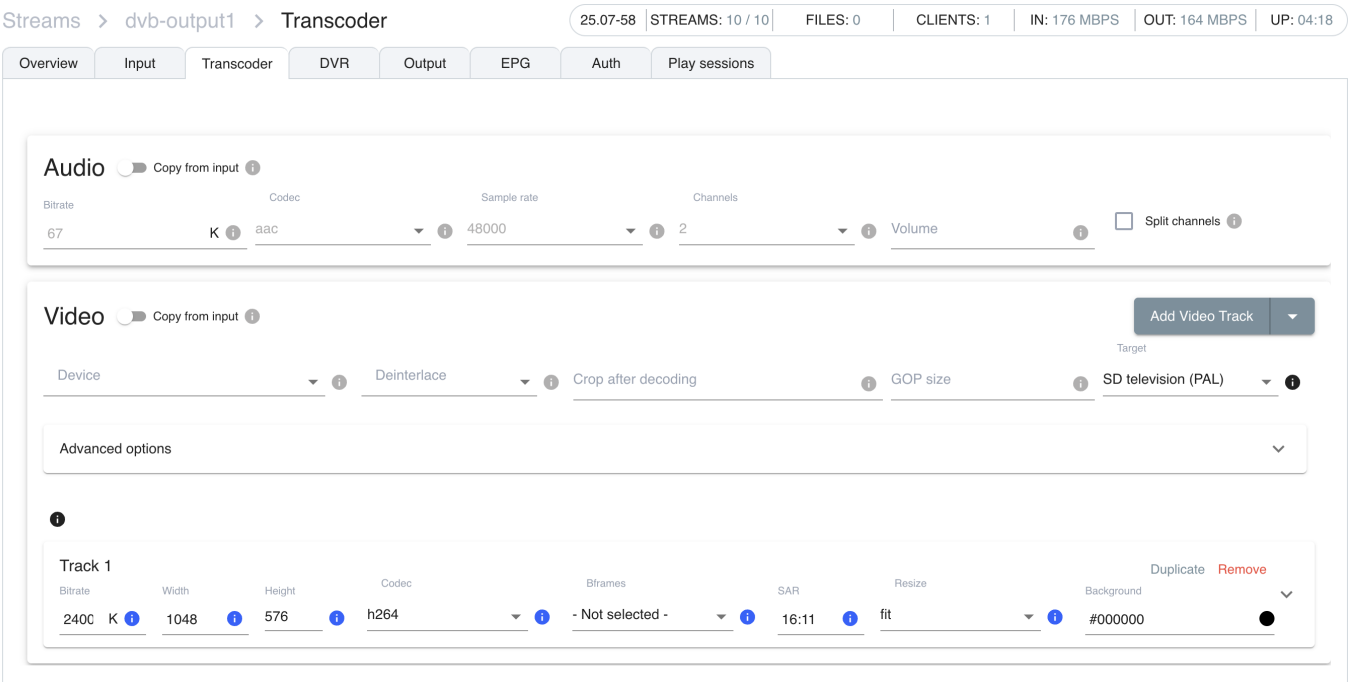
Transmitting Unknown Quality Stream to DVB

- **Problem**: Unstable quality of incoming stream
- **Solution**: Normalization and stabilization of parameters

- **Result**: Guaranteed quality for DVB broadcasting

Operating Modes

DVB CBR (CONSTANT BIT RATE)



Purpose

Mode designed for digital television broadcasting with fixed bitrate, ensuring stable transmission quality.

Configuration

```
stream ort {
  input udp://239.0.0.1:1234;
  transcoder external=false gop=28 hw=cpu vb=6100k bframes=3 vcodec=h264 b-pyramid=strict bufsize=6000k rc-lookahead=30 x264opts=videoformat=component:no-scenecut:force-cfr:colorprim=bt470bg:transfer=bt470bg:colormatrix=bt470bg:weightb=0 interlace=true open_gop=true preset=fast refs=4 size=1920x1080:scale ab=192k acodec=mp2a atrack=1;
}
```

Key DVB Parameters

Parameter	Description	Value
`vb`	Video bitrate	6100k
`size`	Resolution	1920x1080
`gop`	Group of pictures size	28
`bframes`	Number of B-frames	3
`interlace`	Interlaced scanning	true
`ab`	Audio bitrate	192k
`acodec`	Audio codec	mp2a

OTT MBR (MULTI-BITRATE)

Purpose

Mode designed for internet broadcasting with adaptive bitrate, ensuring optimal quality for various network conditions.

Configuration

```
stream ort {
  input udp://239.0.0.1:1234;
  transcoder vb=600k size=x360 vb=2000k size=x720 vb=5000k size=x2180;
}
```

Key OTT Parameters

Parameter	Description	Value
`vb`	Video bitrate	600k, 2000k, 5000k
`size`	Resolution	x360, x720, x2180

Detailed Configuration Parameters

VIDEO PARAMETERS

Basic Settings

- `external=false` — use built-in transcoder
- `hw=cpu` — hardware acceleration (cpu/gpu)
- `vcodec=h264` — video codec (h264/h265)
- `size=1920x1080` — output video resolution

Quality Parameters

- `vb=6100k` — video bitrate
- `bufsize=6000k` — encoding buffer size
- `rc-lookahead=30` — bitrate control analysis depth
- `preset=fast` — encoding preset (fast/medium/slow)

GOP Parameters

- `gop=28` — group of pictures size
- `bframes=3` — number of B-frames
- `b-pyramid=strict` — B-frame pyramid
- `open_gop=true` — open GOP structure
- `refs=4` — number of reference frames

AUDIO PARAMETERS

- `ab=192k` — audio bitrate
- `acodec=mp2a` — audio codec (mp2a/aac)
- `atrack=1` — number of audio tracks

DVB-SPECIFIC PARAMETERS

- `interlace=true` — interlaced scanning
- `x264opts=videoformat=component` — video format
- `colorprim=bt470bg` — color space
- `transfer=bt470bg` — gamma correction
- `colormatrix=bt470bg` — color matrix

Limitations and Features

MODE INCOMPATIBILITY

Important: It's impossible to prepare a stream simultaneously for DVB and OTT. It's necessary to create two separate streams:

```
# DVB stream
stream dvb_channel {
  input udp://239.0.0.1:1234;
  transcoder vb=6100k size=1920x1080 gop=28 interlace=true;
  output udp://239.0.0.2:1234;
}

# OTT stream
stream ott_channel {
  input udp://239.0.0.1:1234;
  transcoder vb=600k size=x360 vb=2000k size=x720 vb=5000k size=x2180;
  output hls:///var/www/hls/ott;
}
```

TRANSCODER ADAPTABILITY

The module efficiently handles:

- **Source switching** — automatic adaptation to new source
- **Resolution changes** — dynamic encoding parameter adjustment
- **Codec changes** — automatic switching between formats
- **Output stability** — maintaining fixed output parameters

Configuration Examples

HD DVB BROADCASTING

```
stream hd_dvb {
  input udp://239.0.0.1:1234;
  transcoder external=false gop=28 hw=cpu vb=8000k bframes=3 vcodec=h264 b-pyramid=strict bufsize=8000k rc-lookahead=30 x264opts=videoformat=component:no-
scenecut:force-cfr:colorprim=bt709:transfer=bt709:colormatrix=bt709:weightb=0 interlace=false open_gop=true preset=fast refs=4 size=1920x1080:scale ab=256k
acodec=mp2a atrack=2;
  push udp://239.0.0.2:1234;
}
```

OTT MULTIBITRATE

```
stream ott_multibitrate {
  input udp://239.0.0.1:1234;
  transcoder vb=400k size=x240 vb=800k size=x360 vb=1500k size=x480 vb=2500k size=x720 vb=4000k size=x1080;
}
```

SD DVB BROADCASTING

```
stream sd_dvb {
  input udp://239.0.0.1:1234;
  transcoder external=false gop=25 hw=cpu vb=4000k bframes=2 vcodec=h264 b-pyramid=strict bufsize=4000k rc-lookahead=25 x264opts=videoformat=component:no-
scenecut:force-cfr:colorprim=bt470bg:transfer=bt470bg:colormatrix=bt470bg:weightb=0 interlace=true open_gop=true preset=fast refs=3 size=720x576:scale ab=128k
acodec=mp2a atrack=1;
  push udp://239.0.0.3:1234;
}
```

LSI INTEGRATION

```
stream resilient_dvb {
  input copy://primary_source source_timeout=10;
  input copy://backup_source;
  title "Resilient DVB Channel";

  transcoder external=false gop=28 hw=cpu vb=6000k bframes=3 vcodec=h264 b-pyramid=strict bufsize=6000k rc-lookahead=30 x264opts=videoformat=component:no-
scenecut:force-cfr:colorprim=bt709:transfer=bt709:colormatrix=bt709:weightb=0 interlace=false open_gop=true preset=fast refs=4 size=1920x1080:scale ab=192k
acodec=mp2a atrack=1;

  push udp://239.0.0.4:1234;
}
```

Troubleshooting

QUALITY ISSUES

Low Output Stream Quality

1. **Increase bitrate** (`vb`) for better quality
2. **Check quality** of incoming stream
3. **Optimize encoding settings**
4. **Monitor quality metrics**

Unstable Bitrate

1. **Check settings** for `bufsize` and `rc-lookahead`
2. **Ensure stability** of incoming stream
3. **Check system load**
4. **Optimize encoding preset**

PERFORMANCE ISSUES

High CPU Load

1. **Enable hardware acceleration** (`hw=gpu`)
2. **Simplify encoding settings**
3. **Use faster preset** (`preset=veryfast`)
4. **Reduce resolution** or bitrate

Encoding Delays

1. **Reduce** `rc-lookahead` to decrease delay
2. **Optimize GOP size** for quality and delay balance
3. **Check system performance**
4. **Consider using** external transcoder

Configuration Recommendations

DVB OPTIMIZATION

- **Use CBR** for stable bitrate
- **Configure correct color spaces** for standard
- **Optimize GOP** for decoder compatibility
- **Check compliance** with DVB standards

OTT OPTIMIZATION

- **Create multiple bitrates** for adaptability
- **Use progressive scanning** for web players
- **Optimize for mobile devices** (low bitrates)
- **Test on various devices**

GENERAL RECOMMENDATIONS

- **Monitor quality** in real-time
- **Optimize settings** for specific content
- **Use hardware acceleration** when possible
- **Plan redundancy** for critical streams

Conclusion

The Transcoder module provides flexible and efficient video stream transcoding for various broadcasting scenarios. Support for both DVB CBR and OTT MBR modes makes it a universal solution for modern broadcasting systems. Adaptability to incoming stream changes and stability of output parameters ensure reliable operation in complex broadcasting conditions.

3.12.2 DVB-compliant CBR

Taking any supported source, Transcoder can prepare the SPTS to send to DVB network that requires video signal to fit into the constant bitrate bandwidth. The stream is transcoded on the CPU and packaged in MPEG-TS in compliance with ETSI TR 101 290.

> **Warning**
>
> NVENC transcoding doesn't output stable bitrate (CBR) sufficient for DVB standard requirements. We only offer encoding on the CPU.

Suppose you need to prepare SPTS with Full HD resolution and a total bitrate of 6 700 Kbps, the audio PID — 192 Kbps and the video — 6 100 Kbps from an HLS stream. Here, kilobit is 1000 bits, not 1024. You can configure the Flussonic Media Server via web interface or configuration file.

In the web interface

STEP 1. CONFIGURE THE STREAM TRANSCODING TO CBR

1) At the **Streams** page, open the stream settings by clicking on the stream name.

2) Move to the **Transcoder** tab and click **Enable Transcoder**.

3) In the **Video** section, specify `HD Television` as **Target**.

4) Apply the settings by clicking **Save**.

STEP 2. SEND THE STREAM TO A MULTICAST GROUP

1) In the stream settings, go to the **Output** tab in the **Push live video to certain URLs** section and specify the multicast group address as follows: `udp://239.172.0.1:1234`.

2) Apply the settings by clicking **Save**:

In the configuration file

STEP 1. CONFIGURE THE STREAM TRANSCODING TO CBR

In the stream settings, add `transcoder` and configure it as follows:

```
stream spts-cbr {
  input file://vod/bunny.mp4;
  transcoder target=hdtv;
}
```

Here `target=hdtv` will automatically enable 1920x1080 output with 3mbit on video pid.

STEP 2. SEND THE STREAM TO A MULTICAST GROUP

In the stream settings, add `push` and specify the multicast group address: `push udp://239.172.0.1:1234`.

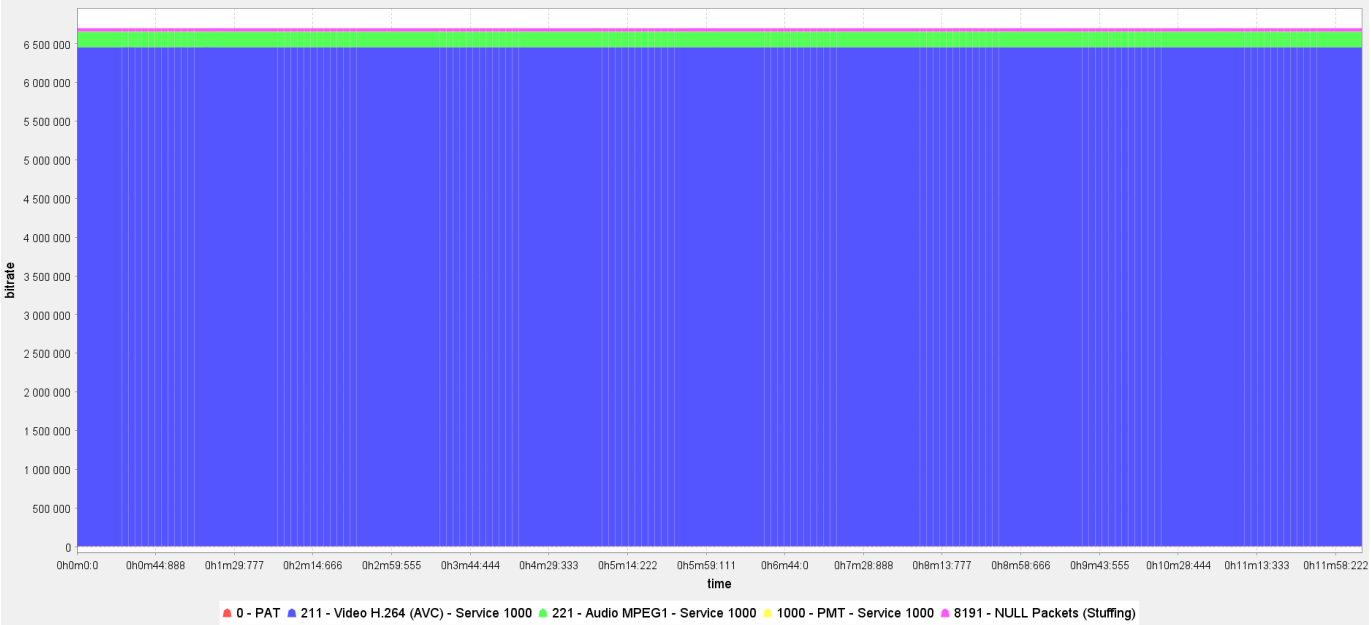
Step 3. Checking the stream quality in the DVB Inspector

Use a tool, such as DVB Inspector (see Checking the stream quality in the DVB Inspector) or any other analyzer that checks the stream for compliance with the ETSI TR 101 290 standard.

1) Record a couple of minutes of a stream using the following command on the terminal: `/opt/flussonic/contrib/multicast_capture.erl udp://239.172.0.1:1234 spts-cbr-output.ts`. Finish recording by pressing *Ctrl+C*.

2) Download the resulting segment `spts-cbr-output.ts` to the local machine.

3) Check the stream segment quality in the в DVB Inspector.



You will get the SPTS at a constant bit rate (CBR) that can be transmitted:

- to a QAM modulator or scrambler to further be sent to the cable network
- to a multiplexer to prepare MPTS

Note

The default step value in DVB Inspector (**View** > **Filter** > **Steps**) smooths out the bitrate of separate PIDs over long durations of segments. It means that the longer the duration of the recorded stream segment, the smoother becomes the bitrate of separate PIDs. For instance, if you record 10 minutes of a stream and open it in the DVB Inspector, you will see a smooth graph with consistent bitrates for separate PIDs. If you set the step to 500 on the same segment, you will notice slight fluctuations in the bitrates of separate PIDs. These minor fluctuations (around one Kbps) don't affect the outcome, and the stream remains CBR.

3.13 SCTE Processor

3.13.1 Overview

SCTE Processor is a module within mcaster that ensures seamless integration of advertising markers in SCTE35 and SCTE104 formats. The module works automatically without additional configuration and solves tasks of converting between different advertising marker formats, as well as compensating for time marker distortions that occur in television paths.

3.13.2 Supported Formats

SCTE35

- **Application**: MPEG-TS, HLS, DASH and other containers with compressed video
- **Structure**: Binary format with PSI/SI tables
- **Location**: Embedded in transport stream

SCTE104

- **Application**: SDI streams
- **Structure**: UDP/IP-based protocol
- **Location**: Separate data stream

3.13.3 Operating Principles

Automatic Conversion

The module automatically detects the format of incoming markers and performs conversion: - **SCTE35 → SCTE104**: For SDI outputs - **SCTE104 → SCTE35**: For MPEG-TS/HLS/DASH outputs

Time Distortion Compensation

SCTE Processor compensates for various types of time marker distortions:

DISTORTIONS IN TELEVISION PATHS

- **Problem**: Time markers in SDI or MPEG-TS may be corrupted
- **Result**: Advertising markers point to non-existent frames
- **Solution**: Automatic compensation and correction of markers

TRANSCODING DISTORTIONS

- **Problem**: Transcoding subsystem changes time markers
- **Result**: Shift in advertising boundaries
- **Solution**: Compensation for changes while maintaining accuracy

CHANGES IN LSI MODULE

- **Problem**: LSI changes time markers when switching sources
- **Result**: Loss of advertising marker synchronization
- **Solution**: Automatic time marker alignment

3.13.4 Processing Architecture

Workflow Diagram

```
Input Stream → SCTE Detection → Format Conversion → Time Compensation → Output Stream
     ↓              ↓                ↓                    ↓              ↓
  SCTE35/104    Identify Type    Convert Format    Fix Timestamps    SCTE35/104
```

Processing Stages

1. **Marker Detection** — automatic identification of SCTE35 or SCTE104 format
2. **Format Conversion** — conversion between SCTE35 and SCTE104
3. **Time Compensation** — correction of distorted time markers
4. **Validation** — verification of processed marker correctness
5. **Insertion** — placement of markers in output stream

3.13.5 Automatic Operation

Out-of-the-Box Operation

The module requires no additional configuration and works automatically: - **Auto-detection** of marker formats - **Auto-conversion** between formats - **Auto-compensation** of time distortions - **Auto-validation** of results

Integration with Other Modules

SCTE Processor automatically integrates with: - **Transcoder** — compensates for time marker changes - **LSI module** — aligns markers when switching sources - **Multiplexer** — ensures correct marker insertion

3.13.6 Technical Features

Time Compensation Algorithms

ADAPTIVE COMPENSATION

- **Analysis of patterns** in time distortions
- **Prediction** of future changes
- **Dynamic adjustment** of markers

MARKER VALIDATION

- **Synchronization check** with video stream
- **Verification of boundaries** for advertising blocks
- **Data integrity control**

Performance

- **Minimal processing delay**
- **High throughput**
- **Low resource consumption**

3.13.7 Monitoring and Diagnostics

Operation Metrics

```
{
  "stats": {
    "input": {
        "ad_splices_ingested": 1250 // Number of processed SCTE markers
```

```
    }
  }
}
```

3.13.8 Recommendations

Operation Optimization

- **Stable sources** — provide more accurate time markers
- **Quality SDI cables** — reduce signal distortions
- **Proper transcoder configuration** — minimize time changes
- **Metrics monitoring** — allows early detection of problems

Parameter Monitoring

- `stats.input.ad_splices_ingested` — number of processed markers

3.13.9 Conclusion

SCTE Processor provides reliable and accurate processing of advertising markers in the mcaster system. The module's automatic operation, intelligent time distortion compensation, and seamless integration with other system components make it an indispensable tool for professional broadcasting with advertising inserts.

3.14 SDI Decoder

The SDI decoder module is designed to send video to the SDI path, received from any other source: SDI, NDI, ST2110, compressed video from DVB or internet.

3.14.1 Simple Configuration

Configuration can be quite simple:

```
stream test {
  input udp://239.0.0.1:1234;
  push dektec://2174220025:2 video_format=pal;
}
```

The Dektec card identifier is obtained in the same way as for the SDI coder module.

3.14.2 Embedded Audio in Television

The module supports embedding radio into television:

```
stream Decoder_Rossia24 {
  input mixer://Fed_asi_board1_port4_PLP1_T2MI_MUX1_1070,Fed_asi_board1_port3_PLP0_T2MI_MUX1_1120,Fed_asi_board1_port3_PLP0_T2MI_MUX1_1110 mixer_strategy=all
sync=dts;
  meta comment "Decoding Rossiya-24+Mayak+VestiFM for GTRK";
  push {
    url dektec://2174223190:2;
    push_audio_tracks {
      channels 1,2;
      sample_type pcm;
      track a1;
    }
    push_audio_tracks {
      channels 3,4;
      sample_type pcm;
      track a2;
    }
    push_audio_tracks {
      channels 5,6;
      sample_type pcm;
      track a3;
    }
    video_format pal;
  }
}
```

3.14.3 Supported Input Sources

The SDI decoder module can receive video from the following sources:

- SDI streams
- NDI streams
- ST2110 streams
- Compressed video from DVB
- Internet streams

3.14.4 Audio Configuration

When configuring audio embedding, you can specify:

- Number of channels for each track
- Sample type (PCM)
- Track number
- Video format (PAL, NTSC, etc.)

3.14.5 Hardware Identification

To work with Dektec cards, you need to specify their identifier in the format:

```
dektec://[ID]:[PORT]
```

Where: * `ID` - unique card identifier * `PORT` - output port number

3.14.6 Decklink SDI Output

Mcaster can not only capture but also output streams to Decklink SDI or HDMI capture cards.

Basic Decklink Output Configuration

For output to Decklink, specify the `push decklink://` parameter:

```
stream test {
  input udp://239.0.0.1:1234;
  push decklink://0;
}
```

Mcaster decodes and then transmits the stream to the specified device number or port on the card (e.g., 0). If necessary, you can specify the `deinterlace=true` option to eliminate interlacing.

Decklink Card Modes

Usually, a Decklink card supports a limited set of modes. Each mode is a combination of frame size and frame rate, encoded in Decklink format. For example, 1080i50 means frame size 1920x1080 and frame rate 50000/1000 FPS. When sending a stream to a Decklink card, you can set the mode value in the `format` parameter:

```
stream test {
  input udp://239.0.0.1:1234;
  push decklink://0 format=1080i50;
}
```

Possible Decklink operating modes are described in the API documentation.

Duplex Mode Operation

To specify Duplex mode for a DeckLink SDI card, allowing you to choose input and output direction, use the following configuration in global DeckLink settings:

```
decklink {
  duplex_mode two_half;
}
```

The example above shows semi-duplex mode configuration for a DeckLink Duo 2 card.

DeckLink Quad 2 and DeckLink Duo 2 cards have non-standard numbering when mapping physical and logical ports, which affects duplex mode configuration.

To configure a DeckLink Quad 2 card so that all ports are used either as input or output, the card should operate in `two_half` mode:

```
decklink {
  duplex_mode two_half;
}
```

3.14.7 VBI Teletext Output (Important for Legacy SD Systems)

VBI teletext output is a critically important feature for legacy SD broadcasting systems that still transmit analog teletext. Mcaster can pass teletext from MPEG-TS to analog streams in SD quality that are broadcast via Decklink or DekTec SDI cards. Teletext is added to VBI (vertical blanking interval) of an output stream.

Prerequisites

- An input MPEG-TS stream containing **Teletext B**
- An output stream containing SD video that Mcaster will transmit to a Decklink or DekTec SDI card

VBI Teletext Configuration

To pass a teletext track to SDI, specify numbers of the lines where the teletext in the output stream will be packed:

```
stream out {
  input file://vod/mpegts.ts;
  push decklink://1 format=pal vbi_lines=ttxt:7:8:9:319:320:321;
}
```

In the example, the `vbi_lines` option specifies six figures separated by colons — these are numbers of VBI lines that will carry a teletext track. The first three are VBI lines passed in the first half-frame and the next three figures are lines in the second half-frame.

Important Notes

- If the teletext in your stream does not fit into the specified lines, it will not appear in the output stream
- In this case, specify more lines in `vbi_lines`
- This feature is essential for integration with legacy SD broadcasting systems that continue to use analog teletext
- VBI teletext is particularly important for maintaining compatibility with older television equipment

> **Note**
>
> This functionality is especially critical for legacy SD broadcasting systems that still transmit analog teletext and require VBI integration for proper operation.

3.15 ASI Push

The ASI push module as part of Mcaster not only captures MPTS streams but also passes them to DekTec ASI cards for output to ASI interfaces.

3.15.1 Purpose

This module is designed for:

- Outputting MPTS streams to ASI interfaces
- Integration with broadcast equipment
- Professional video distribution systems
- Cable and satellite headend applications

3.15.2 Configuration

To enable pushing to ASI, add the pusher configuration like `push dektec-asi://serial_number:port` to the multiplexer:

```
transponder mux1 {
  bitrate 10000k;
  push dektec-asi://2174207373:3;
  program 1 {
    source channel_1;
    title test;
    lcn 0;
    service_type digital_tv_avc_hd;
    pid 5032 pmt  pmt;
    pid 5033 a1;
    pid 5034 v1 bitrate=6000 pcr;
    pid 5035 l1;
  }
}
stream channel_1 {
  input fake://fake;
}
```

3.15.3 Hardware Requirements

For ASI output, DekTec ASI cards are required. They provide:

- Professional-grade ASI interfaces
- High reliability for broadcast applications
- Support for various bitrates
- Multiple output ports

3.15.4 Supported Formats

The module supports:

- MPTS (Multi-Program Transport Stream) output
- Various video codecs (H.264, H.265)
- Multiple audio formats
- Different bitrates up to 270 Mbps

3.15.5 Configuration Parameters

DekTec Card Identification

The DekTec card is identified by:

- **serial_number** - unique card identifier
- **port** - output port number

Stream Configuration

Each program in the multiplexer can be configured with:

- **source** - input stream name
- **title** - program title
- **lcn** - logical channel number
- **service_type** - type of service
- **pid** - program identifier with options

3.15.6 Applications

ASI push is used in various scenarios:

- Broadcast television distribution
- Cable headend systems
- Satellite uplink facilities
- Professional video production
- Content distribution networks

3.15.7 Monitoring

The module provides monitoring capabilities for:

- Output stream status
- Bitrate monitoring
- Error detection
- Hardware status

3.16 Multiplexer

Multiplexer is the most important element of the entire system for delivering television to DVB networks.

3.16.1 Main Tasks

It covers the following tasks:

- Preparing MPEG-TS stream in compliance with DVB standard requirements for CBR, PCR accuracy, HRD buffer, etc.
- Packaging multiple TV channels into MPTS while maintaining uniform channel interleaving
- Forming all service tables: PAT, PMT, SDT, EIT, NIT necessary for a full DVB service

3.16.2 Configuration Example

```
transponder plp0 {
  provider "Central TV";
  bitrate 22600k;
  ts_stream_id 1;
  network 0x3578 original=0x2283 name="DTT - National TV";
  time_offset RUS:10 time_of_change=2018-03-23T03:00:00Z local_time_offset=+0300 next_time_offset=+0300;
  version psi 25;
  ts_descriptor 0x7F 040022830325;
  push udp://streaming0@238.238.31.1:1111 multicast_loop tos=0;
  program 1010 {
    source Switcher_1010_ort;
    title "01 ORT";
    eit_title Ort_(+0);
    lcn 1;
    service_type digital_tv_avc_sd;
    pid 1010 pmt;
    pid 1011 v1 pcr bitrate=2720 ;
    pid 1012 a1 ;
    pid 1014 l1 codec=ttxt  ;
    pid 1015 l1 codec=scte35  ;
    pid 1016 bypass   es_info=6F030010E1 stream_type=5;
    pid 1017 bypass   es_info=140D000B00000080000000FFFFFFFF52010B130500002F520066020123 stream_type=11;
    pid 1018 bypass   es_info=52010D stream_type=12;
  }
  program 1040 {
    source Switcher_1040_04_Uchebniy;
    title "04 Educational";
    eit_title Uchebniy_(+0);
    lcn 4;
    service_type digital_tv_avc_sd;
    pid 1040 pmt;
    pid 1041 v1 pcr bitrate=2750 ;
    pid 1042 a1;
    pid 1044 l1 codec=ttxt  ;
    pid 1045 l1 codec=scte35  ;
    pid 1046 bypass   es_info=6F030010E1 stream_type=5;
    pid 1047 bypass   es_info=140D000B00000080000000FFFFFFFF52010B130500002F520066020123 stream_type=11;
    pid 1048 bypass   es_info=52010D stream_type=12;
  }
  other @plp1;
  eit {
    source bypass://EIT_source_MUX1_plp0;
    max_bitrate 300;
  }
}
```

3.16.3 Configuration Details

Program Sections

In the multiplexer configuration, `program` sections are specified - these are individual TV channels. They must reference a stream configured in Mcaster.

Program Parameters

- **title** - TV channel name
- **eit_title** - optional title that goes to EPG (Electronic Program Guide)

- **lcn** (Logical Channel Number) - sequential number in the program list, transmitted in NIT table
- **service_type** - service type, specified in SDT table

PID Configuration

The PID list is rigid, fixed, not dynamic and will not change depending on the source.

PID OPTIONS

- **pmt** - PMT table will go in this PID
- **pcr** - PCR (Program Clock Reference) will be marked on this PID
- **codec** - you can separately specify the PID codec
- **bitrate** - you can force the PID bitrate
- **bypass** - PIDs that are passed through without demultiplexing at the input

BYPASS PID PARAMETERS

For bypass PIDs, you can specify MPEG-TS packaging parameters:

- **es_info** - elementary stream information
- **stream_type** - stream type

Additional Parameters

- **other** - allows referencing other multiplexers for SDT formation
- **eit** - EIT table formation:
- From file
- From stream where program schedule is captured

3.16.4 Monitoring Parameters

The module provides the following parameters for monitoring:

Main Metrics

- **payload** - payload bytes count (multiplexer_payload)
- **encoded** - encoded bytes count (multiplexer_encoded)
- **fillers** - filler packets count (multiplexer_fillers)
- **stuffing** - stuffing packets count (multiplexer_stuffing)

Additional Metrics

- **trimmed_bytes** - trimmed bytes count (multiplexer_trimmed_bytes)
- **trimmed_frames** - trimmed frames count (multiplexer_trimmed_frames)
- **ts_overflow** - TS stream overflow (boolean)

3.16.5 Service Tables

Multiplexer forms all necessary DVB service tables:

- **PAT** (Program Association Table) - program association table
- **PMT** (Program Map Table) - program map table
- **SDT** (Service Description Table) - service description table
- **EIT** (Event Information Table) - event information table
- **NIT** (Network Information Table) - network information table

3.16.6 DVB Standard Requirements

The module ensures compliance with the following requirements:

- **CBR** (Constant Bit Rate) - constant transmission rate
- **PCR accuracy** - Program Clock Reference accuracy
- **HRD buffer** - hypothetical decoder buffer requirements
- Uniform channel interleaving in multiplex

3.16.7 Electronic Program Guide (EPG)

Mcaster can generate MPTS streams with an embedded electronic program guide (EPG). You no longer need to use an additional EPG generator and remultiplexer to add EPG to TV channels.

In MPEG-TS streams, the EPG is stored in the Event Information Tables (EIT). Mcaster can populate EIT in the output stream.

Two Ways to Add EPG

There are two ways for Mcaster to add EPG to the output MPTS:

1. **Copy EIT from source** if the program guide in it is satisfiable
2. **Take the EPG from XMLTV files and convert it into EIT** for both current multiplexer (Actual) and other multiplexers in a network (Other)

Copy EPG from Source

To copy EIT from the source MPEG-TS stream, add EIT options to the multiplexer configuration:

```
transponder tp1 {
  eit {
    source copy://STREAMNAME;
  }
}
```

Replace the `STREAMNAME` with the name of the source stream in Mcaster configuration.

Import EPG from XMLTV

To import EPG from XMLTV files, configure the multiplexer as follows:

```
transponder tp1 {
  program 100 {
    title "program1";
    eit_title "example_title";
  };
  other @tp2;
  eit {
    xmltv_url xmltv_dir1;
    interval pf actual=1 other=2;
    interval schedule other=20;
  }
}
```

CONFIGURATION PARAMETERS

- `title` — sets the channel ID value from the `channel id` in the XMLTV file
- `eit_title` — sets the channel name from the `display-name` in XMLTV
- `xmltv_url` — sets the path to the directory with XMLTV files (can be a single file)
- `interval pf|schedule actual=<INTERVAL 1> other=<INTERVAL 2>` — sets how often EIT tables will be sent

EPG RELOADING

When EPG data is updated in the XMLTV file, reload it using the API:

```
POST /streamer/api/v3/multiplexers/{name}/xmltv_upload
```

IMPORTANT NOTES

- The EPG is packed into the target bitrate
- EIT version changes when schedule is updated (number from 0 to 63)
- XMLTV files might contain overlapping transmissions - Mcaster includes the earlier one
- Default intervals: `actual` (present/following) is 2 seconds, `other` (present/following) is 4 seconds, `actual` (schedule) and `other` (schedule) are 60 seconds

Note

The description of the XMLTV file format can be found at xmltv.org, and broadcast schedules in this format are available at teleguide.info.

3.17 TwinCast Recovery

The TwinCast Recovery module provides automatic failover functionality for multicast streaming with zero downtime. It ensures maximum reliability for content delivery by automatically switching between primary and backup servers without interrupting the broadcast.

3.17.1 Overview

TwinCast Recovery is designed for content providers who need to deliver multicast streams with maximum reliability. The module implements a sophisticated standby mechanism that monitors the primary server and automatically activates the backup server when needed.

Key Features

- **Zero Downtime Failover**: Automatic switching without broadcast interruption
- **Standby Mode**: Backup servers monitor primary server status
- **IGMP Compatibility**: Works with any IGMP receiver
- **Automatic Recovery**: Seamless return to primary server when available
- **Real-time Monitoring**: Continuous status monitoring via `stats.push[0].standby_status`

3.17.2 How Standby Mode Works

The standby mode operates through a sophisticated monitoring and switching mechanism:

Primary Server Operation

1. **Active Broadcasting**: Primary server sends streams to multicast group address
2. **Continuous Monitoring**: Backup server constantly monitors multicast group activity
3. **Status Verification**: Backup server checks if any server is sending multicast packets

Backup Server Operation

1. **Standby State**: Backup server remains in standby mode while primary is active
2. **Automatic Activation**: When primary server goes offline, backup starts streaming
3. **Automatic Deactivation**: When primary returns, backup stops and returns to standby

Failover Process

- **Detection**: Backup server detects absence of multicast packets from primary
- **Activation**: Backup server immediately starts streaming to multicast group
- **Recovery**: When primary server returns, backup automatically stops and returns to standby
- **Monitoring**: Continuous monitoring ensures immediate response to any issues

3.17.3 Configuration

Basic Setup

Configure TwinCast Recovery through the Mcaster Admin UI:

1. **Access Stream Settings**: Go to the **Output** tab of stream settings
2. **Configure Primary Server**: In the *URL* column of **Push live video to certain URLs** section, specify: `udp://239.1.1.1:1234` Where:
3. `239.1.1.1` - multicast group address
4. `1234` - port for Mcaster to listen to
5. **Add Backup Server**: Configure backup server with same multicast group settings
6. **Enable Standby Mode**: Open **Options** and enable standby mode by checking *Standby* box
7. **Apply Changes**: Click **Save** to activate configuration

Advanced Configuration

MULTICAST GROUP SETTINGS

```
stream primary_stream {
  input udp://239.0.0.1:1234;
  push udp://239.1.1.1:1234;
}

stream backup_stream {
  input udp://239.0.0.1:1234;
  push udp://239.1.1.1:1234 standby;
}
```

NETWORK CONSIDERATIONS

- **Multicast Routing**: Ensure proper multicast routing configuration
- **IGMP Snooping**: Configure switches for optimal multicast delivery
- **Bandwidth Planning**: Account for both primary and backup streams
- **Network Isolation**: Consider VLAN configuration for multicast traffic

3.17.4 Monitoring and Status

Standby Status Monitoring

Monitor the standby status using the `stats.push[0].standby_status` field:

```
{
  "stats": {
    "push": [
      {
        "standby_status": "active|waiting"
      }
    ]
  }
}
```

Status Values

- **active**: Server is currently broadcasting to multicast group
- **waiting**: Server is monitoring but not broadcasting (standby mode)

Monitoring Best Practices

- **Regular Status Checks**: Monitor `standby_status` field regularly
- **Log Analysis**: Review logs for failover events
- **Performance Metrics**: Track switching times and reliability

- **Alert Configuration**: Set up alerts for status changes

3.17.5 Use Cases

Content Provider Redundancy

- **Satellite Broadcasting**: Ensure continuous satellite feed delivery
- **Cable Networks**: Maintain service during server maintenance
- **Live Events**: Prevent broadcast interruption during critical events
- **24/7 Operations**: Maintain service availability around the clock

Enterprise Applications

- **Corporate Broadcasting**: Reliable internal video distribution
- **Educational Institutions**: Uninterrupted lecture streaming
- **Government Communications**: Reliable emergency broadcast systems

3.17.6 Troubleshooting

Common Issues

BACKUP SERVER NOT ACTIVATING

- **Check Network Connectivity**: Verify multicast routing
- **Verify Standby Configuration**: Ensure standby mode is enabled
- **Monitor Logs**: Check for error messages in system logs
- **Test Multicast Group**: Verify multicast group accessibility

PRIMARY SERVER NOT DETECTED

- **Network Configuration**: Check multicast group settings
- **Firewall Rules**: Ensure multicast traffic is allowed
- **IGMP Configuration**: Verify IGMP snooping on switches
- **Server Status**: Confirm primary server is broadcasting

FREQUENT FAILOVERS

- **Network Stability**: Check for network instability
- **Server Performance**: Monitor primary server resources
- **Multicast Congestion**: Check for multicast traffic issues
- **Configuration Conflicts**: Verify no conflicting multicast groups

Diagnostic Commands

```
# Check multicast group status
netstat -g

# Monitor multicast traffic
tcpdump -i eth0 udp port 1234

# Verify IGMP membership
cat /proc/net/igmp
```

3.17.7 Performance Optimization

Switching Speed Optimization

- **Network Latency**: Minimize network latency between servers

- **Detection Time**: Optimize multicast packet detection timing
- **Buffer Configuration**: Adjust buffer sizes for optimal performance
- **CPU Resources**: Ensure adequate CPU resources for monitoring

Resource Management

- **Memory Usage**: Monitor memory consumption during failover
- **CPU Utilization**: Track CPU usage during standby monitoring
- **Network Bandwidth**: Plan for multicast traffic requirements
- **Storage I/O**: Consider storage requirements for logging

3.17.8 Security Considerations

Network Security

- **Multicast Authentication**: Implement multicast authentication if required
- **Network Isolation**: Use VLANs to isolate multicast traffic
- **Access Control**: Restrict access to multicast configuration
- **Monitoring**: Monitor for unauthorized multicast traffic

Configuration Security

- **Secure Configuration**: Protect configuration files
- **Access Logging**: Log all configuration changes
- **Backup Security**: Secure backup of configuration files
- **Update Procedures**: Follow secure update procedures

3.18 RTMP Pusher

Mcaster allows you to publish any stream to an external server using RTMP protocol.

Social media platforms use RTMP to organize live broadcasts, which means you can use Mcaster to send your streams to social media (it can be several at once).

3.18.1 Use Cases

- Receiving video from a mobile reporter and sending directly to several social media platforms
- Broadcasting video from CCTV cameras
- Broadcasting own programs in social media, including scheduled broadcasts

> **Warning**
>
> Stream keys can expire. Check the terms of the service before publishing the stream.

3.18.2 Publish to YouTube

To publish a stream to YouTube:

1. Go to YouTube Studio and create a new live stream
2. Copy the stream URL and stream key
3. In Mcaster configuration, add RTMP push settings
4. Configure the stream to push to YouTube's RTMP server

3.18.3 Publish to Facebook

To publish a stream to Facebook:

1. Go to Facebook Live and create a new live stream
2. Copy the server URL and stream key
3. In Mcaster configuration, add RTMP push settings
4. Configure the stream to push to Facebook's RTMP server

3.18.4 Publish to OK

To publish a stream to OK.ru:

1. Go to OK.ru broadcast section and create a new stream
2. Copy the server URL and broadcast key
3. In Mcaster configuration, add RTMP push settings
4. Configure the stream to push to OK.ru's RTMP server

3.18.5 Configuration Example

```
stream youtube_stream {
  input udp://239.0.0.1:1234;
  push rtmp://a.rtmp.youtube.com/live2/YOUR_STREAM_KEY;
}

stream facebook_stream {
  input udp://239.0.0.1:1234;
  push rtmp://live-api.facebook.com:80/rtmp/YOUR_STREAM_KEY;
```

```
}

stream ok_stream {
  input udp://239.0.0.1:1234;
  push rtmp://vsu.mycdn.me/input/YOUR_STREAM_KEY;
}
```

3.18.6 Multiple Destinations

You can push the same stream to multiple destinations simultaneously:

```
stream multi_social {
  input udp://239.0.0.1:1234;
  push rtmp://a.rtmp.youtube.com/live2/YOUTUBE_KEY;
  push rtmp://live-api.facebook.com:80/rtmp/FACEBOOK_KEY;
  push rtmp://vsu.mycdn.me/input/OK_KEY;
}
```

3.18.7 Monitoring

Mcaster provides monitoring capabilities for RTMP push operations:

- Connection status to each destination
- Push statistics and metrics
- Error reporting for failed connections
- Automatic reconnection on connection loss

3.19 SRT Egress

Mcaster supports sending video streams via SRT protocol. SRT (Secure Reliable Transport) is widely used for delivering video over the Internet or satellite networks, as it guarantees low latency while offering content delivery guarantees.

3.19.1 Overview

SRT Egress module allows you to push streams from Mcaster to external servers using SRT protocol. This is particularly useful for:

- Delivering video content to remote servers
- Broadcasting to CDN networks
- Satellite transmission
- Low-latency video distribution

3.19.2 Basic Configuration

To configure SRT egress, use the `push` directive in your stream configuration:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  push srt://destination-server.com:9998 streamid="#!::r=stream-name,m=publish";
}
```

3.19.3 URL Formats

SRT push URLs can be configured in two formats:

SRT Parameters in URL Parameters

```
srt://SRT-HOST:SRT_PORT streamid="#!::r=STREAM_NAME,m=publish"
```

SRT Parameters in URL Query String

```
srt://SRT-HOST:SRT_PORT?streamid=#!::r=STREAM_NAME,m=publish
```

Where: * `SRT-HOST` - IP address of the destination server * `SRT_PORT` - SRT port number * `STREAM_NAME` - name of the publishing location on the destination server

3.19.4 Configuration Examples

Simple SRT Push

```
stream srt_output {
  input udp://239.0.0.1:1234;
  push srt://example.com:9998 streamid="#!::r=my-stream,m=publish";
}
```

SRT Push with Parameters

```
stream srt_secure {
  input udp://239.0.0.1:1234;
  push srt://example.com:9998?streamid=#!::r=secure-stream&passphrase=1234567890;
}
```

Multiple SRT Destinations

```
stream multi_srt {
  input udp://239.0.0.1:1234;
  push srt://server1.com:9998 streamid="#!::r=stream1,m=publish";
  push srt://server2.com:9999 streamid="#!::r=stream2,m=publish";
  push srt://server3.com:10000?streamid=#!::r=stream3&passphrase=secret123;
}
```

3.19.5 SRT Parameters

You can configure various SRT parameters for optimal performance:

Security Parameters

- `passphrase` - encryption passphrase for secure transmission
- `pbkeylen` - public key length for encryption

Performance Parameters

- `latency` - maximum latency tolerance
- `rcvbuf` - receive buffer size
- `sndbuf` - send buffer size
- `mss` - maximum segment size

Example with Parameters

```
stream optimized_srt {
  input udp://239.0.0.1:1234;
  push srt://example.com:9998?streamid=#!::r=optimized-stream&passphrase=secret&latency=120&rcvbuf=8192&sndbuf=8192;
}
```

3.19.6 Stream ID Format

The `streamid` parameter follows a specific format:

```
#!::r=STREAM_NAME,m=publish
```

Where: * `r=STREAM_NAME` - specifies the stream name on the destination server * `m=publish` - specifies the mode (publish for sending)

3.19.7 Use Cases

CDN Distribution

```
stream cdn_output {
  input udp://239.0.0.1:1234;
  push srt://cdn-provider.com:9998 streamid="#!::r=live-channel,m=publish";
}
```

Satellite Transmission

```
stream satellite {
  input udp://239.0.0.1:1234;
  push srt://satellite-gateway.com:9998?streamid=#!::r=broadcast&passphrase=satellite-key;
}
```

Remote Studio

```
stream remote_studio {
  input udp://239.0.0.1:1234;
```

```
  push srt://studio-server.com:9998 streamid="#!::r=studio-feed,m=publish";
}
```

3.19.8 Error Handling

SRT Egress module provides automatic error handling:

- Automatic reconnection on connection loss
- Retry mechanisms for failed connections
- Error logging for troubleshooting
- Graceful degradation when destination is unavailable

3.19.9 SRT Playback

Mcaster also supports playing SRT streams from the server. This allows clients to receive video streams via SRT protocol.

Basic SRT Playback Configuration

To configure SRT playback for a single stream, use the `srt_play` block in your stream configuration:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  srt_play {
    port 9998;
  }
}
```

To play the stream, clients use the following URL format:

```
srt://SERVER-IP:SRT_PORT
```

Where: * `SERVER-IP` - IP address of your Mcaster server * `SRT_PORT` - SRT port specified for playback

For the example above, the playback URL would be: `srt://localhost:9998`

Combined Publish and Play

You can configure a single SRT port for both publishing and playing a stream:

```
stream example_stream {
  input publish://;
  srt 9998;
}
```

For playback, use the following URL format:

```
srt://SERVER-IP:SRT_PORT?streamid=#!::m=request
```

Where: * `m=request` - specifies playback mode

The URL for this example would be: `srt://localhost:9998?streamid=#!::m=request`

Global SRT Playback Port

To enable one SRT port for playing multiple streams, use `srt_play` as a global setting:

```
srt_play {
  port 9998;
}
stream example_stream {
  input udp://239.0.0.1:1234;
}
stream another_stream {
  input udp://239.0.0.1:1235;
}
```

To play streams over the global port, use the following URL format:

```
srt://SERVER-IP:SRT_PORT?streamid=#!::r=STREAM_NAME
```

Where: * `r=STREAM_NAME` - specifies the stream name

For the example above: * `example_stream`: `srt://localhost:9998?streamid=#!::r=example_stream` * `another_stream`: `srt://localhost:9998?streamid=#!::r=another_stream`

SRT Playback with Parameters

You can configure SRT playback with additional parameters:

```
stream secure_stream {
  input udp://239.0.0.1:1234;
  srt_play {
    port 9998;
    passphrase 0987654321;
  }
}
```

The playback URL with parameters:

```
srt://SERVER-IP:9998?passphrase=0987654321&streamid=#!::m=request
```

URL Format Summary

Playback URL	Configuration	Description
`srt://SERVER-IP:PORT`	`srt_play { port PORT; }`	Single stream per port
`srt://SERVER-IP:PORT?streamid=#!::m=request`	`srt PORT;`	Combined publish and play
`srt://SERVER-IP:PORT?streamid=#!::r=STREAM_NAME`	Global `srt_play`	Multiple streams per port
`srt://SERVER-IP:PORT?streamid=#!::r=STREAM_NAME,m=request`	Global `srt_play` + `srt_publish`	Global port with publish streams

3.20 OTT Packager

Mcaster OTT Packager is designed to prepare transcoded content for playback on mobile devices and players. It processes video streams after the transcoder module to create adaptive streaming formats suitable for various devices and network conditions.

3.20.1 Overview

The OTT Packager module takes transcoded video streams and packages them into industry-standard adaptive streaming formats. This enables efficient content delivery to mobile devices, smart TVs, and web browsers with optimal quality based on available bandwidth.

Key Features

- **Multiple Format Support**: HLS, DASH, and MSS protocols
- **Adaptive Bitrate Streaming**: Automatic quality selection based on network conditions
- **CDN Integration**: Works with Flussonic CDN and other CDN providers
- **Mobile Optimization**: Optimized for mobile device playback
- **Archive Support**: Handles both live and archived content

3.20.2 Supported Protocols

HLS (HTTP Live Streaming)

HLS is Apple's adaptive streaming protocol, widely supported across devices and platforms.

Advantages:

- Universal compatibility with iOS, Android, and web browsers
- Automatic quality adaptation
- Simple implementation and debugging

Use Cases:

- Mobile applications
- Web browsers
- Smart TVs and set-top boxes

DASH (Dynamic Adaptive Streaming over HTTP)

DASH is an international standard for adaptive streaming, offering high efficiency and flexibility.

Advantages:

- ISO standard with broad industry support
- Efficient bandwidth utilization
- Advanced features for live and VOD content

Use Cases:

- High-quality streaming services
- Multi-platform applications
- Professional broadcasting

MSS (Microsoft Smooth Streaming)

MSS is Microsoft's adaptive streaming technology, optimized for Windows platforms.

Advantages:

- Excellent Windows ecosystem integration
- Advanced DRM support
- Professional-grade features

Use Cases:

- Windows applications
- Xbox platforms
- Enterprise streaming solutions

3.20.3 Configuration

Basic Configuration

```
stream ott_output {
  input udp://239.0.0.1:1234;
  segment_duration 2;
  segments 10;
}
```

Where: * `segment_duration` - segment duration in seconds * `segments` - number of segments kept in memory and available in playlist

3.20.4 Playback URLs

After configuring the OTT Packager, you can play streams using the following URLs:

HLS Playback

```
http://streamer/{stream_name}/index.m3u8
```

DASH Playback

```
http://streamer/{stream_name}/Manifest.mpd
```

MSS Playback

```
http://streamer/{stream_name}.isml/manifest
```

Where `{stream_name}` is the name of your configured stream.

3.20.5 CDN Integration

Flussonic CDN

We recommend building CDN infrastructure using Flussonic technologies for optimal integration and performance.

Benefits:

- Seamless integration with Mcaster OTT Packager
- Optimized performance and reliability
- Comprehensive monitoring and analytics
- Advanced caching and edge delivery

Third-Party CDN

Mcaster OTT Packager is compatible with other CDN providers, allowing flexibility in infrastructure choices.

Supported CDN Types:

- Traditional CDN providers
- Cloud-based CDN services
- Hybrid CDN solutions

3.20.6 Archive Playback

Sliding Window Mode

When playing archived content, separation between CDN and Packager may not be possible. In such cases, archive playback servers need to operate in sliding window mode.

Key Considerations:

- Continuous segment generation
- Dynamic playlist updates
- Efficient storage management
- Real-time content availability

DVR Integration

For comprehensive archive playback functionality, refer to the DVR section for detailed information about:

- Archive storage configuration
- Playback server setup
- Sliding window implementation
- Performance optimization

3.20.7 Performance Optimization

Segment Optimization

- **Segment Length**: Balance between latency and efficiency
- **Playlist Length**: Optimize for memory usage and startup time
- **Bitrate Ladders**: Configure appropriate quality levels

Caching Strategy

- **CDN Caching**: Leverage CDN edge caching for improved delivery
- **Local Caching**: Implement local caching for frequently accessed content
- **Cache Headers**: Configure appropriate cache control headers

3.20.8 Monitoring and Analytics

Key Metrics

- **Segment Generation Rate**: Monitor segment creation performance
- **Playlist Update Frequency**: Track playlist generation efficiency
- **CDN Delivery Performance**: Measure content delivery metrics

- **Client Playback Statistics**: Analyze viewer behavior and quality

Health Monitoring

- **Service Availability**: Monitor packager service status
- **Error Rates**: Track packaging and delivery errors
- **Resource Utilization**: Monitor CPU, memory, and storage usage
- **Network Performance**: Analyze bandwidth and latency metrics

3.20.9 Security Considerations

Content Protection

- **DRM Integration**: Support for various DRM systems
- **Token Authentication**: Secure access control
- **Geographic Restrictions**: Content geo-blocking capabilities
- **Encryption**: Secure content transmission

Access Control

- **Authentication Methods**: Various authentication mechanisms
- **Authorization Levels**: Role-based access control
- **Session Management**: Secure session handling
- **Audit Logging**: Comprehensive access logging

3.20.10 Troubleshooting

Common Issues

- **Segment Generation Failures**: Check transcoder output and storage permissions
- **Playlist Errors**: Verify configuration and file system access
- **CDN Delivery Problems**: Monitor network connectivity and CDN status
- **Client Playback Issues**: Analyze client compatibility and network conditions

Debug Tools

- **Log Analysis**: Comprehensive logging for issue identification
- **Performance Monitoring**: Real-time performance metrics
- **Configuration Validation**: Automated configuration checking
- **Health Checks**: Automated service health monitoring

3.20.11 Subtitles and Teletext Support

Teletext Processing

Mcaster OTT Packager can recognize DVB subtitles, read teletext and closed captions in MPEG-TS and pass them to HLS and DASH.

Teletext Support:

- **VBI (SD SDI)**: Data located in the invisible area of the frame
- **OP-47 (HD SDI)**: Teletext specification for HD SDI allowing more stable transmission
- **Automatic Conversion**: Teletext is automatically converted to WebVTT and TTML formats

Supported Cards:

Card	VBI (SD SDI)	OP-47 (HD SDI)
DekTec		
Decklink		
Stream Labs		
Magewell		
AJA		

Closed Captions

CEA-608/708 Support:

- **CEA-608**: Standard for analog closed captions (channels 1-4)
- **CEA-708**: Standard for digital closed captions (channels 1-63)
- **Automatic Extraction**: Closed captions are automatically extracted and converted

Configuration Example:

```
stream example_stream {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.extract;
  substyle valign=top align=left;
}
```

Caption Signaling:

To enable closed captions in manifests, add parameters to the input URL:

```
stream example_stream {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.708.12.lang=fr cc.608.1.lang=eng;
}
```

Where: * `cc.708.12.lang=fr` - CEA-708 channel 12 in French * `cc.608.1.lang=eng` - CEA-608 channel 1 in English

Subtitle Formats

HLS Support:

- **WebVTT**: Primary subtitle format for HLS
- **Automatic Conversion**: Teletext and closed captions automatically converted to WebVTT

DASH Support:

- **WebVTT**: Web Video Text Tracks format
- **TTML**: Timed Text Markup Language (default format)
- **Format Selection**: Choose format via URL parameter `?text=wvtt` or `?text=ttml`

MSS Support:

- **TTML**: Primary subtitle format for MSS
- **Automatic Processing**: Any subtitle type converted to TTML format

Subtitle Positioning

Configure subtitle position using the `substyle` parameter:

```
stream example_stream {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.extract;
```

```
  substyle valign=top align=left;
}
```

Positioning Options:

- **Vertical Alignment**: `valign=top|middle|bottom`
- **Horizontal Alignment**: `align=left|center|right`

TTML Subtitles

TTML (Timed Text Markup Language) is a standard for closed captioning and subtitling that offers:

- **Rich Features**: Positioning, alignment, styling, multiple languages
- **Industry Standard**: Widely supported by media players and streaming platforms
- **XML-based**: Text file with .ttml or .xml extension
- **Professional Grade**: Used in television industry

Features:

- Multiple language support
- Advanced styling options
- Precise timing control
- Accessibility compliance

3.21 QAM Output

Mcaster allows you to push streams to ATSC-C cable network without need to use additional modulation devices. In this case, a TBS card (currently we support TBS6014) works as a signal generator for modulation of an MPTS (multiplexer), i.e. converting it to QAM signal that can be transported to cable networks for TV broadcasting.

3.21.1 Hardware Requirements

To configure QAM output, you need a TBS card (currently TBS6014 is supported) that works as a signal generator for modulation.

3.21.2 DVB Card Configuration

To configure such a pusher, first add to the configuration a DVB card with the following necessary parameters:

- `hw` – the device model, its value should be `tbs6014`
- `adapter` – adapter number
- `port` – port number

Example:

```
dvb_card tbsmod01 {
  hw tbs6014;
  adapter 0;
  frequency 62000000;
  modulation qam256;
  interleave 3;
  gain 5;
  port 0;
  input_bitrate 38;
}
```

Optional Parameters

- `frequency` — the carrier frequency (MHz) of the multiplexer for this channel
- `modulation` — TBS modulation method
- `interleave` — use interleaver. The interleaver disperses sequence of bits in bit stream to minimize effect of burst errors during transmission
- `gain` — adjust the output gain to the specified value in dB
- `input_bitrate` — input bitrate, in Mbps

3.21.3 Multiplexer Configuration

Then configure a multiplexer with the option `push dvb://tbsmod01`. For example:

```
stream channel1 {
  input udp://239.0.0.1:1234;
}
stream channel2 {
  input udp://239.0.0.2:1234;
}
transponder newMultiplexer1 {
  bitrate 26970k;
  push dvb://tbsmod01;
  program 100 {
    source channel1;
    title Channel1;
    lcn 0;
    service_type digital_tv_avc_sd;
    pid 101 pmt  pmt;
    pid 102 v1 bitrate=2000 pcr ;
    pid 103 a1 bitrate=500  ;
  }
  program 200 {
    source channel2;
    title Channel2;
    lcn 1;
    service_type digital_tv_avc_sd;
    pid 201 pmt  pmt;
```

```
    pid 202 v1 bitrate=500 pcr ;
    pid 203 a1 bitrate=100  ;
  }
}
```

3.21.4 Modulation Methods

When choosing the multiplexer bitrate, keep in mind the used modulation method because it can limit the ability to accept the data:

- **QAM64** — maximal possible bitrate 26.90735 Mbit/s
- **QAM256** — maximal possible bitrate 38.81070 Mbit/s

3.21.5 Applications

QAM output is used in various scenarios:

- Cable television networks
- ATSC-C broadcasting
- Local cable headend systems
- Professional video distribution
- Multi-channel broadcasting

3.21.6 Supported Standards

The module supports:

- ATSC-C cable standards
- QAM modulation (64-QAM, 256-QAM)
- MPTS (Multi-Program Transport Stream) output
- Professional cable network integration

3.22 Qprober

3.22.1 Overview

Qprober is an integrated monitoring system within all Mcaster components that measures various event counts across sources and individual data streams within sources, such as MPEG-TS PIDs. The module provides comprehensive stream quality analysis and real-time problem diagnostics.

3.22.2 System Architecture

Component Integration

Qprober is integrated into all Mcaster components:

- **Input modules** — monitoring sources and streams
- **Processing modules** — analysis of processing quality
- **Output modules** — control of output stream status
- **System components** — monitoring server resources

Centralized Monitoring

Measured counters can be collected centrally across an entire server cluster, but when centralized system is organizationally impossible, Mcaster Appliance package may include built-in visualization of all counters.

3.22.3 Main Metrics

Basic Server Indicators

- **CPU load** — processor resource usage
- **Disk load** — I/O operations and performance
- **GPU load** — graphics processor usage
- **Memory usage** — RAM and swap consumption

General Video Stream Indicators

- **Bitrate** — current and average stream bitrates
- **Source status** — availability and quality of sources
- **Frame rate** — FPS of input and output streams
- **Resolution** — current resolution of video streams

3.22.4 Error Monitoring

Source Unavailability

Qprober tracks:

- **Connection loss** with sources
- **Connection timeouts**
- **Authentication errors**
- **Network problems** to source

Instrumental MPEG-TS Stream Analysis

Analysis according to TR101290 standard includes:

PRIORITY 1 (CRITICAL ERRORS)

- **Synchronization** — stream synchronization loss
- **PAT** — Program Association Table errors
- **PMT** — Program Map Table errors
- **CC** — Continuity Counter errors

PRIORITY 2 (IMPORTANT ERRORS)

- **PCR** — Program Clock Reference errors
- **PTS/DTS** — timestamp errors
- **CAT** — Conditional Access Table errors

PRIORITY 3 (INFORMATIONAL ERRORS)

- **NIT** — Network Information Table errors
- **SDT** — Service Description Table errors
- **EIT** — Event Information Table errors

Network Protocol Analysis

SRT ANALYSIS

- **Frame loss** — number of lost frames
- **Emergency timestamp jumps** — sharp time changes
- **RTT** — Round Trip Time of connection
- **Retransmissions** — number of retransmitted packets

RTSP ANALYSIS

- **Packet loss** — RTP packet loss statistics
- **Jitter** — delay variations
- **Session errors** — RTSP session problems

RTMP ANALYSIS

- **Frame loss** — loss statistics
- **Protocol errors** — RTMP connection problems
- **Buffering** — buffer status

3.22.5 Output Stream Monitoring

Internal Problems

Qprober tracks:

- **Encoding errors** — transcoder problems
- **Multiplexing problems** — multiplexer errors
- **Buffering errors** — buffer overflow

Response to Input Problems

The module records:

- **Timestamp correction** — automatic time corrections
- **Counter reset** — when drift accumulates
- **Source switching** — automatic switching to backup

Examples of Tracked Events

```
{
  "timestamp": "2024-01-15T10:30:00Z",
  "source": "input_stream_1",
  "event": "timestamp_correction",
  "details": {
    "drift_accumulated": 1500,
    "correction_applied": 1500,
    "counters_reset": true
  }
}
```

3.22.6 Problem Diagnostics

Determining Problem Source

Qprober allows determining the source of incoming problems:

NETWORK PROBLEMS

- **High RTT** in SRT connections
- **Packet loss** in RTMP/RTSP
- **Unstable bitrate**
- **Frequent reconnections**

SOURCE PROBLEMS

- **TR101290 errors** in MPEG-TS
- **Synchronization loss**
- **Incorrect timestamps**
- **Encoding problems**

Metrics for Analysis

The main metrics that need to be studied are obtained through the `streams_list` API. The `streams` object contains a list of all streams, where each stream has a `stats` object with more than a hundred parameters for analysis.

GETTING METRICS

```
# Getting list of all streams with metrics
curl -X GET "http://localhost:8080/api/streams_list"
```

RESPONSE STRUCTURE

```
{
  "streams": [
    {
      "name": "main_stream",
      "stats": {
        // More than 100 parameters for analysis
        "input": {
          "packets_received": 125000,
          "packets_lost": 5,
          "bitrate": 5000000,
          "fps": 25.0,
          "tr101290": {
            "priority1_errors": 0,
            "priority2_errors": 2,
            "priority3_errors": 5
```

```
          },
          "srt": {
            "rtt": 25.5,
            "retransmitted_packets": 15,
            "latency": 35.2
          }
        }
      }
    }
  ]
}
```

KEY METRIC GROUPS

Input metrics (`stats.input`): - **packets_received/lost** — packet statistics - **bitrate/fps** — stream quality - **tr101290** — MPEG-TS errors by standard - **srt/rtmp/rtsp** — protocol-specific metrics

Output metrics (`stats.output`): - **packets_sent** — sent packets - **bitrate/fps** — output stream quality - **errors** — encoding/multiplexing errors

System metrics (`stats.system`): - **cpu_usage** — processor load - **memory_usage** — memory usage - **disk_io** — input-output operations

3.22.7 Retroview Integration

Online Mode

It is recommended to use Qprober in online mode together with Retroview service:

- **Real-time** — instant problem analysis
- **Historical data** — long-term trend analysis
- **Automatic alerts** — problem notifications
- **Centralized monitoring** — single control point

Offline Mode

Can also be used in offline on-premises mode:

- **Local storage** — data remains in infrastructure
- **Autonomous operation** — independence from external services
- **Built-in visualization** — as part of Mcaster Appliance

3.22.8 Configuration

Basic Settings

```
qprober {
    enabled true;
    sampling_interval 1000;  # milliseconds
    retention_period 86400;  # seconds
    log_level info;
}
```

Stream Monitoring Configuration

```
stream monitored_stream {
    input udp://239.0.0.1:1234;

    qprober {
        tr101290_analysis true;
        network_metrics true;
        output_metrics true;
        alert_threshold 10;
    }
}
```

Configuration Parameters

Parameter	Description	Required	Example
`enabled`	Enable Qprober	Yes	`true`
`sampling_interval`	Metrics collection interval	No	`1000`
`retention_period`	Data retention period	No	`86400`
`tr101290_analysis`	TR101290 analysis	No	`true`
`network_metrics`	Network metrics	No	`true`
`alert_threshold`	Alert threshold	No	`10`

3.22.9 API and Interfaces

HTTP API

```
# Getting stream metrics
GET /api/qprober/stream/stream_name

# Getting system metrics
GET /api/qprober/system

# Getting TR101290 errors
GET /api/qprober/tr101290/stream_name

# Getting network metrics
GET /api/qprober/network/stream_name
```

Prometheus Metrics

```
# System metrics
mcaster_cpu_usage{server="server1"} 45.2
mcaster_memory_usage{server="server1"} 67.8
mcaster_disk_io{server="server1"} 125.5

# Stream metrics
mcaster_stream_bitrate{stream="main",server="server1"} 5000000
mcaster_stream_fps{stream="main",server="server1"} 25.0

# TR101290 errors
mcaster_tr101290_priority1{stream="main",server="server1"} 0
mcaster_tr101290_priority2{stream="main",server="server1"} 2
mcaster_tr101290_priority3{stream="main",server="server1"} 5
```

3.22.10 Data Visualization

Built-in Visualization

Mcaster Appliance includes built-in visualization:

- **Dashboards** — real-time key indicators
- **Graphs** — historical data and trends
- **Alerts** — critical event notifications
- **Reports** — detailed analytics

External System Integration

Qprober supports integration with:

- **Grafana** — for advanced visualization
- **Prometheus** — for metrics collection
- **ELK Stack** — for log analysis
- **Zabbix** — for infrastructure monitoring

3.22.11 Usage Recommendations

Performance Optimization

- **Configure collection interval** for metrics according to your needs
- **Use filtering** to reduce load
- **Plan storage** of historical data
- **Monitor resources** of Qprober itself

Alert Configuration

- **Define critical thresholds** for your streams
- **Configure escalation** of notifications
- **Use different levels** of alert importance
- **Test alerts** in test environment

Data Analysis

- **Regularly analyze** quality trends
- **Correlate problems** with external factors
- **Document** typical problems and solutions
- **Plan improvements** based on data

3.22.12 Conclusion

Qprober represents a powerful monitoring and stream quality analysis system integrated into all Mcaster components. The module provides comprehensive problem diagnostics, from basic system metrics to detailed MPEG-TS stream analysis according to TR101290 standard. The ability for centralized monitoring of server clusters and built-in visualization make Qprober an indispensable tool for ensuring broadcast quality in professional systems.

4. Standards

4.1 TR 101 290

The development of digital television (not as an abstract term, but as a specific set of specifications and products collectively referred to as DVB) coincided with an era where governmental regulators were able to keep pace with market developments and establish standards that all players had to follow to operate in the market.

To establish common service provision rules and measurable quality indicators, DVB developed a set of *instrumental* checks that demonstrate the correctness of byte stream formation in MPEG-TS. This document is known as ETSI TR 101 290.

In TR101290, it is clearly and unambiguously described how to specifically check the byte stream from the recipient to consider the television stream suitable for *quality* analysis. Picture quality analysis is done using other methods, with key terms being PSNR, SSIM, and VMAF. The document has been adopted by regulators in various countries as mandatory and serves as a good example of how regulatory work can improve the overall market condition by setting clear rules in the form of an unambiguously interpretable document.

The checks themselves are computationally simple enough to allow constant monitoring of hundreds and thousands of channels without difficulty. This differs from quality checks; for example, VMAF is rarely run constantly and is typically tested selectively, for just a few minutes per hour.

An analogous document might be the one used to write Apple's mediastreamvalidator for the HLS protocol or an extended XML+MP4 validator for DASH.

The part of TR101290 that describes the bytes themselves consists of three chapters, sorted by criticality, called priorities. In addition to this part, there are descriptions of checks in various environments, such as DVB-T/T2, DVB-C/S.

4.1.1 Usage of the standard

The document itself is important only for digital television (DVB) and primarily for non-IP environments, such as cable, terrestrial, and satellite. In these contexts, it is very meaningful and significant.

However, it is not an integral part of MPEG-TS itself because its requirements are entirely meaningless in the context of HLS, for example. In HLS, PCR is not used at all, which would be considered heresy for a typical television broadcaster.

4.1.2 1 priority

Section **5.2.1 First priority: necessary for de-codability (basic monitoring)** describes the most critical issues that prevent the transport stream from being unpacked. These issues are:

- `TS_sync_loss`, `Sync_byte_error`: inability to synchronize with byte 0x47
- `PAT_error`, `PAT_error_2`: the absence of PAT on PID 0 or its regular loss (information containing the list of TV channels in the stream)
- `PMT_error`, `PMT_error_2`: insufficiently frequent repetition of PMT tables on the PIDs listed in the PAT (description of each TV channel)
- `PID_error`: some PIDs are declared, but no packets are received on them within the desired time (the standard here is flexible, leaving the choice to the user)
- `Continuity_count_error`: CC errors, which actually indicate loss, duplication, or reordering of packets, with loss being the most common.

In practice, the last point often leads to misunderstandings between individuals with television experience and those with programming experience.

For programmers, the phrase "CC error" primarily sounds like "the code generates an invalid byte stream," whereas it could very well mean "packets are lost due to microbursts, resulting in insufficient data on the receiver's end."

4.1.3 2 priority

The next section, **5.2.2 Second priority: recommended for continuous or periodic monitoring**, describes the correctness of data generation by the program that created the byte stream. In the DVB world, the concept of remultiplexing is often used, where transport streams are not fully unpacked to frames and back, but are partially rewritten, keeping some data unchanged. Errors can occur during these repacking processes, which are also

tracked by this part of the document. Additionally, any of the errors listed below (and above) may arise due to bits changing during transmission, causing the receiver to see something different from what the source sent.

- `Transport_error` : an upstream program has set the indicator for a broken stream. This can be used to raise an alert only in monitoring without disrupting playback on televisions.
- `CRC_error` : one of the service tables was modified, but the checksum was not recalculated.
- `PCR_error` : the interval between consecutive stream time stamps is too large.
- `PCR_repetition_error` : time stamps are marked too infrequently and need to be more frequent.
- `PCR_discontinuity_indicator_error` : there was a jump in the stream time, but the discontinuity indicator (or source switching indicator) was not set.
- `PCR_accuracy_error` : the most mysterious error for those who think PCR is related to real time. It refers to the non-uniformity of stream time stamping.
- `PTS_error` : PTS is set too infrequently. This error dates back to when it was possible not to timestamp frames with PTS/DTS in MPEG-TS at all. In HLS, nothing will play without these markers, but PCR can be omitted.
- `CAT_error` : a problem with stream decryption.

4.1.4 PCR

PCR, also known as stream time, is one of the most persistent myths, especially in light of the intimidating requirement of 500-nanosecond accuracy. It is often mistakenly associated with absolutely precise timing, leading to myths that a computer cannot generate PCR, while a magical hardware multiplexer made on very expensive Israeli (why not?) FPGA cards can, due to the need for precise timing.

PCR is simply a packet number recalculated using a linear formula: `PCR = PCR_initial + (N*188*8) / Bitrate`.

That's all the magic; feel free to use it. The trick is that this formula only makes sense when the stream is prepared with CBR (Constant Bit Rate) requirements, meaning the same number of bytes arrives every second. For HLS, these requirements do not exist, so there is no point in talking about PCR in HLS.

Our media server, Flussonic, generates the MPEG-TS stream independently, timestamps everything itself, packs it in CBR, and delivers a zero-jitter PCR.

4.1.5 3 priority

- `NIT_error` , `NIT_actual_error` , `NIT_other_error` , `SI_repetition_error` , `TDT_error` , `RST_error` , `SDT_other_error` : These checks pertain to the frequency and correctness of informational packets (tables) that describe the transport stream and its contents.
- `Unreferenced_PID` : This indicates that a PID exists but is not listed in any table. This can happen when extraneous PIDs are added and then extracted on the other side, essentially replacing the instrumental stream structure with voice-over.
- `Buffer_error` , `Empty_buffer_error` : Errors related to the HRD buffer, which are classified under the third priority, although they probably belong in the second priority.
- `EIT_error` , `EIT_actual_error` , `EIT_other_error` , `EIT_PF_error` : These errors are related to the schedule, ensuring it arrives on time and is up-to-date.

4.1.6 HRD Buffer error

Finding details about this simple mechanism isn't easy, so we'll explain it here.

A frame with a specific PTS (indicated in the PES header of the frame) starts being transmitted to the receiver, which accumulates it in a buffer. The buffer grows with each incoming packet. Once the stream time PCR exceeds the specified PTS, the frame is entirely removed from the buffer, reducing its fill level. This results in a sawtooth graph pattern.

The buffer should not exceed the upper limit (predetermined) or empty "below zero," meaning the frame must arrive completely by the time the specified stream time is reached. The transcoder is responsible for compressing the frame sufficiently to meet this requirement.

Buffer depletion is one of the few issues that significantly affect picture quality because the television either has to formally decode a broken frame or wait for the complete frame to arrive, resulting in a jerky picture.

4.2 Digital TV broadcasting

Digital television developed alongside the Internet, replacing the analog signal. Digital TV supported features that are unusual for a conventional Internet connection. That's why DVB (Digital Video Broadcasting) systems developed necessary solutions at the time and remain relevant but are useless for the Internet.

This article describes the features of DVB, including distribution over IP networks (IPTV), regards to how it differs from OTT delivery and UGC services, such as YouTube or Twitch.

There are several competivive standards of digital television: DVB, ATSC, ISDB-T. They are rather similar to each other, differ in details a bit.

All of them are describing how to use MPEG-TS to transfer TV channel with strict compliance to TR 101 290

1. DVB application
2. DVB in IP
3. Noninteractive DVB
4. Engineer aspects of DVB
5. MPEG-TS
    a. PID and CC
    b. PAT, PMT, and other tables
    c. EIT
    d. CBR encoding
    e. PCR

4.2.1 DVB application areas

The following areas require DVB:

- satellite TV (DVB-S)
- cable TV (DVB-C)
- terrestrial TV (DVB-T)
- cable TV over IP (IPTV). IPTV often stands apart from DVB, but technical and frame organization relates IPTV to DVB.

The broadcasting in these environments is unidirectional. It means that a source prepares the same set of data for each subscriber and broadcasts it. The number of subscribers doesn't affect the source. The source doesn't know the number of subscribers it has. Unlike the Internet, where the number of clients creates a non-linear load on a server.

4.2.2 DVB in IP

IPTV relates to DVB because the concept of IPTV service is the same as in other DVB environments: unidirectional low-interactive distribution of TV channels over a pre-designed proprietary delivery network. IPTV has the same set of services and user devices, such as set-top boxes and TV sets as DVB-C.

The difference between IPTV and DVB is the *HbbTV (Hybrid Broadcast Broadband Television)*. It's a technology that provides interactive services to IPTV.

4.2.3 Interactivity of DVB

Due to unidirectional distribution, DVB lacks interactive services. It means that if consumers of satellite TV services switch off the set-top box for a long time, the set-top box will miss the session of receiving keys to decode the signal. To activate a new set-top box, the technician of such a service contacts the office and asks them to send keys to the satellite. Using UGC and OTT services, viewers can activate their desired service in just a second.

Without interactive services, you risk losing revenue as subscribers can't immediately access a sports package of TV channels to watch events like the World Cup final with friends. To avoid losing revenue and provide interactive services, traditional DVB services transitioned to IP networks and implemented *middleware*. **Middleware** is a website with an embedded player.

4.2.4 Engineer aspects of DVB

Unidirectional DVB delivery systems such as satellite, cable, and terrestrial have the following technical features:

- Constant bitrate.

  For example, a satellite transmitter on the right frequency transmits precisely 48 MB of data per second. If the transmitter software stops responding for a couple of seconds, subscribers will lose the signal. For engineers, this means creating soft real-time systems with strict bitrate stability requirements.

- Push model of services.

  In DVB services, to deliver a set of content or TV schedules, data has to be prepared in a certain way. The data is often sent to consumers, even if they don't need this data. Since the data channel dedicated to services is limited, it's challenging to add new services.

- Regulated protocols and services.

  Committees and government regulators control DVB at the protocol level. It means that any TV works with DVB services, but the list of all possible services is limited and hasn't changed for years. DVB has a set of content as a list of TV channels and a schedule of TV programs on those TV channels in a limited form. The later implementation of HbbTV allowed to provide interactive services and diversity to outdated specifications.

4.2.5 MPEG-TS

DVB uses the MPEG-TS packaging format to deliver television. MPEG-TS isn't a protocol because there is no description of client-server interaction in the standard. Often such interaction doesn't exist.

MPEG-TS is suitable for DVB tasks because it's designed for beyond IP networks. It's when there are no boundaries on the number of packets, addresses, or ports, and a stream of bytes, in which you need to somehow find the boundaries of packets and quickly start showing video.

In practice, MPEG-TS shows that the picture appears on the screen within a couple of seconds.

In the context of DVB, you need to know the following about MPEG-TS:

- What's PID and CC
- PAT, PMT, NIT tables to arrange the list of TV channels
- EIT tables for transmitting EPG
- CBR encoding content
- PCR, which attracts more than enough attention

PID и CC

MPEG-TS was designed to pack several parallel data streams into a single physical channel.

In IP networks, parallel data streams are separated by IP addresses and ports at both ends of the connection. In MPEG-TS, this separation is made by specifying a 13-bit channel number within the shared stream at the beginning of each packet. All packets have the same size: 188 bytes with four-byte headers. In these four bytes, the first byte is reserved for a fixed number 0x47 used to find packet boundaries in the byte stream (three 0x47 bytes running through 187 bytes are sufficient for synchronization). The remaining 3 bytes are reserved for PID (Programme Identifier), *CC (Continuity Counter)*, and other flags.

CC (Continuity Counter) is a 4-bit counter indicating that packets aren't missed. Unlike RTP, DVB doesn't involve packet rearrangement, so you can tell if a pair of packets got lost by using CC. It's more complex to distinguish the loss of one packet and 17 packets.

Under normal conditions, there are no CC errors in the output from the Flussonic Media Server because the server doesn't lose anything anywhere. CC errors can arise due to a restart of the stream or packet loss further downstream.

PAT, PMT, and other tables

PAT, PMT, and other tables establish the transmission format of the list of TV channels available on the current and neighboring frequencies. DVB standard is under government committees control. It means that all set-top boxes and TV sets can read it. For decades, DVB hasn't experienced any development or change.

The standards of the committees don't keep up with the market needs. The situation with LCN (Logical Channel Number) proves the point.

To establish the order of TV channels for consumers, providers use different solutions and none of them has become a standard. It means that for different TV sets, the order is different.

The allocated PIDs transfer records of TV channels in bits. Such design isn't scalable, but vendors and operators come up with their solutions to overcome this complication. Their solutions work and even support adding new codecs.

Flussonic generates all the tables from scratch rather than passes them through from input to output. This way, you get the correct output stream.

EIT

EIT (Event Information Table) transmits the EPG (Electronic Programme Guide) TV schedule in a separate PID.

The metadata channel is constrained, as evidenced by the design of this feature. The TV schedule for the current day is sent much more often than for future days. There's no rush to receive the schedule if the program is scheduled many hours away, but the set-top box still needs to quickly access the current schedule.

The algorithm for preparing EPG schedules can be complex, often requiring dedicated programs. Flussonic has a built-in EIT generator that uses XMLTV to prepare the EPG. For details on how to add the EPG in Flussonic, see Adding the EPG to MPTS in Mcaster.

CBR encoding

Only audio can achieve true constant bitrate (CBR) as each frame of MPEG-2 audio is encoded using the same number of bytes. CBR video is possible when transferring between 300 and 3000 Mb per stream with raw codec or separate frame compression. While the H.264 codec typically operates in variable bitrate (VBR), you can try to achieve CBR by compressing each frame to the required size.

When analyzing one-hour files of a variable bitrate (VBR) stream, you'll notice that they have consistent sizes. It makes them constant bit rate (CBR) streams but with a larger window. In DVB, this window equals one second. It means that the total number of bytes for every consecutive number of frames (typically 25 FPS) remains within the specified limit.

No encoder achieves the specified bitrate. When examining a stream in a DVB analyzer, you'll see graphs displaying even bitrates, thanks to bit stuffing. MPEG-TS maintains the required traffic by inserting Null packets into the stream. Transmitted over a dedicated port, Null packets don't provide any valid information. H.264 uses NAL units (Network Abstraction Layer Units) for stuffing, which are not displayed by analyzers.

DVB encoders are expected to approach the specified minimum bitrate without exceeding it. However, ensuring such precise guarantees for encoders is often excessive, and few transcoders are designed with such strict requirements in mind.

PCR

PCR (Programme Clock Reference)—time stamps showing the timing embedded into the stream. A media player should use it as a reference. Suppose the player received a frame that has a PTS (Presentation TimeStamp). The player will show the frame when the required PCR arrives in the stream.

PCR doesn't require real-time systems and can be set on an ordinary server, achieving 100% PCR accuracy and zero jitter in the output. **PCR accuracy** tells you how much the linearity of packet number growth and PCR growth don't match. To create a stream that maintains consistent bitrates for separate PIDs even when certain PIDs are skipped, you need a software that meets several requirements from the DVB standard.

Flussonic drops all incoming PCR marks and often ignores them because they aren't required. Then Flussonic inserts the PCR marks in the output stream with zero jitter value.