

DELTACAST NMOS Service Documentation

This documentation corresponds to NMOS Service version 2.0.0.

Table of Contents

- [DELTACAST NMOS Service Documentation](#)
 - [Table of Contents](#)
 - [Overview](#)
 - [What is NMOS?](#)
 - [Supported NMOS interfaces](#)
 - [Service Architecture](#)
 - [Key Capabilities](#)
 - [NMOS Resource Hierarchy](#)
 - [System Requirements](#)
 - [Hardware Requirements](#)
 - [Software Requirements](#)
 - [Supported Operating Systems](#)
 - [Installation](#)
 - [Prerequisites \(All Platforms\)](#)
 - [DNS Service Discovery](#)
 - [Windows](#)
 - [Linux](#)
 - [macOS](#)
 - [PTP](#)
 - [Windows](#)
 - [Linux](#)
 - [Windows Installation](#)
 - [macOS Installation](#)
 - [Linux Installation](#)
 - [Configuration](#)
 - [Configuration File Location](#)
 - [Configuration File Structure](#)
 - [NMOS Configuration Section](#)
 - [Basic Identity Settings](#)
 - [Network Configuration](#)
 - [Security Configuration \(HTTPS/TLS\)](#)
 - [On-Board Configuration](#)
 - [On-Board Configuration Parameters](#)
 - [Service Configuration Section](#)
 - [Operational Settings](#)

- Example Configurations
 - Example 1: Out-of-Board Configuration
 - Example 2: On-Board Configuration
- Applying Configuration Changes
- Configuration Validation
- Working with TX and RX Streams
 - Understanding TX Streams
 - Understanding RX Streams
 - Opting Out of NMOS Exposure
 - Sample Applications
- Command-Line Options
- Support
- Licensing and Legal Information
 - Service License
 - Open Source Software Components
 - Core Dependencies
 - Proprietary Components
 - Transitive Dependencies
 - License Compliance
 - Warranty Disclaimer
- Appendix A: HTTPS Setup on Windows
 - Prerequisites
 - Step 1: Generate Certificates in WSL
 - 1. Install OpenSSL and create the configuration file
 - 2. Generate keys and build the PFX container
 - 3. Transfer files to Windows
 - Step 2: Import Certificates into Windows Certificate Store
 - 1. Install the Root CA certificate
 - 2. Install the Server PFX certificate
 - Step 3: Configure HTTP.SYS SSL Binding
 - 1. Retrieve the certificate thumbprint
 - 2. Create the SSL binding using netsh
 - Step 4: Enable HTTPS in Configuration

Overview

The DELTACAST NMOS Service is a cross-platform system service that offers NMOS (Networked Media Open Specifications) support for DELTACAST ST 2110 PCIe video boards. This service acts as a bridge between the DELTACAST VideoMaster SDK and the NMOS ecosystem, enabling sophisticated IP media workflows in broadcast, production, and post-production environments.

What is NMOS?

NMOS is a family of specifications developed by the Advanced Media Workflow Association (AMWA) that standardizes discovery, connection management, and control of IP media devices. By implementing NMOS, devices from different manufacturers can work together seamlessly in professional media networks.

Supported NMOS interfaces

The DELTACAST NMOS Service supports the following NMOS interfaces:

- **IS-04:** Node registration and resource publication
- **IS-05:** Flow connection and management
- **IS-09:** System-level configuration (e.g. PTP)

These APIs enable discovery, connection management, and configuration of DELTACAST devices in IP media networks. For details on each API, refer to the [AMWA NMOS specifications](#).

Service Architecture

The DELTACAST NMOS Service supports two deployment models:

- **Out-of-board:** The service runs as a system-level daemon on the host computer, managing NMOS functionality for all installed DELTACAST PCIe boards
- **On-board:** NMOS runs directly on the Deltacast.TV device, providing integrated NMOS control enabled and coordinated by this service

Both deployment models support **out-of-band** NMOS control, while on-board deployment additionally supports **in-band** control through the device's embedded network interfaces. Deployment models are exclusive, either on-board or out-of-board NMOS can be active at a time.

Key Capabilities

The service provides comprehensive NMOS functionality:

- **Automatic Discovery:** Automatically detects all installed DELTACAST boards
- **Dynamic Resource Mapping:** Maps RX and TX streams to NMOS resources (Sources, Flows, Senders, Receivers)
- **Connection Management:** Enables remote configuration and connection of TX/RX streams
- **Security:** Supports HTTPS with X.509 certificate-based authentication

NMOS Resource Hierarchy

Understanding the NMOS resource model is essential for working with this service:

- **NMOS Node:** The service instance itself is registered as a single NMOS Node. This Node represents the host computer and acts as a container for all devices and their resources.
- **NMOS Device:** Each physical DELTACAST board installed in the system is mapped to a separate NMOS Device. A Device represents a logical grouping of related sources,

senders, and receivers that share common hardware. NMOS Devices are created dynamically: a Device appears in the registry only when at least one active stream exists on the corresponding board, and is removed when no streams remain.

- **TX Streams** (Transmit): Each active TX stream on a DELTACAST board is represented by a complete set of NMOS resources:
 - **Source**: Represents the origin of the media content (video or audio)
 - **Flow**: Describes the media format, codec, and timing characteristics
 - **Sender**: Represents the network transmission endpoint with transport parameters
 - All TX streams are controllable via IS-05 Connection Management API
- **RX Streams** (Receive): Each RX stream is mapped to:
 - **Receiver**: Represents the network reception endpoint
 - Receivers are controllable via IS-05 for dynamic connection to remote Senders

This hierarchical structure ensures that all resources are properly organized and discoverable by NMOS controllers and other devices on the network.

System Requirements

Hardware Requirements

- One or more DELTACAST ST 2110 PCIe video boards:
 - **DELTA-IP25-44-elp**: Latest generation ST 2110 board with on-board NMOS support
 - **DELTA-IP-ST2110 10**: ST 2110 capture card (out-of-board mode only)
 - **DELTA-IP-ST2110 01**: ST 2110 playout card (out-of-board mode only)

Software Requirements

- DELTACAST VideoMaster SDK (version 6.34.xxx)
- **Windows only**: Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (x64)
 - Download from: https://aka.ms/vs/17/release/vc_redist.x64.exe
- For production environments: PTP (Precision Time Protocol) setup is recommended. The NMOS service relies on system time for scheduled activations and accurate timestamping.

Supported Operating Systems

- **Windows**
- **macOS**
- **Linux**

Installation

Prerequisites (All Platforms)

Before installing the NMOS service, ensure that **DELTACAST VideoMaster redists** are properly installed

DNS Service Discovery

The NMOS service uses DNS Service Discovery (DNS-SD) for automatic discovery of NMOS registries and peer-to-peer operation. The implementation varies by platform:

Windows

The Apple Bonjour service must be installed on Windows systems for DNS-SD support.

Linux

The [Avahi](#) daemon provides DNS-SD support on Linux systems.

To install Avahi and the required compatibility libraries on Ubuntu/Debian:

```
sudo apt-get install -f avahi-daemon libnss-mdns libavahi-compat-libdnssd1
```

The Avahi daemon will be installed and configured to start automatically.

macOS

No need to install anything.

PTP

The system clock should be synchronized to a grandmaster using PTP for accurate timing. Multiple PTP implementations are available for different platforms:

Windows

For Windows, the recommended PTP implementation is the DELTACAST Delta-PTP.

Linux

The Linux PTP project provides a robust PTP implementation for Linux systems. It includes the `ptp4l` daemon for PTP synchronization and `phc2sys` for synchronizing the system clock to the PTP hardware clock.

Windows Installation

The service is distributed as a Windows installer package: `nmos-service-win.x64-<version>.exe`.

Installation Steps:

1. Run the installer and follow the prompts.
2. Go to `C:\ProgramData\DELTACAST\nmos-service` and edit `config.jsonc`. More details in the [Configuration](#) section.
3. The service is installed but not running by default. Use the service interface to start it:
 - Open Services (`services.msc`)
 - Locate `DELTACAST NMOS Service`

- Right-click and select `Start` , set to `Automatic` startup type if desired, ...
- 4. For the HTTP back-end, the service relies on HTTP.SYS and the Windows Certificate Store for managing HTTPS connections and certificates. In the default installer-driven setup, no manual URL ACL step is normally required. Manual URL ACL reservations are only needed if you run the executable outside the installed Windows service, change the service account, or encounter HTTP.SYS namespace reservation errors.

Installation Directories:

- Executable: `C:\Program Files\DELTACAST\nmos-service`
- Configuration and logs: `C:\ProgramData\DELTACAST\nmos-service`

Windows HTTP.SYS Namespace Reservation

The application relies on the Windows HTTP Server API (HTTP.SYS) for HTTP endpoint binding.

In certain system configurations, explicit URL namespace reservations (URL ACLs) may be required for the service account running the application.

If binding fails, create a URL reservation using:

```
netsh http add urlacl url=http://+:PORT/ user=<service_account>
```

PORT should be replaced with the actual port number configured for the NMOS APIs (e.g., 3212 for IS-04, 3215 for IS-05).

Administrative privileges are required.

Proper configuration of HTTP namespace reservations and related Windows networking settings (including WinHTTP policies where applicable) is the responsibility of the deploying organization.

To list existing URL reservations:

```
netsh http show urlacl
```

macOS Installation

1. Run the MacOS installer package: `nmos-sevice-macosx-<Version>.pkg` .
2. Accept the license agreement and follow the installation prompts.
3. After installation, edit the configuration file located at `/etc/nmos-service/config.jsonc` . More details in the [Configuration](#) section.
4. The service is installed as a launchd daemon and can be managed using `launchctl` commands.
5. The logs are in `/var/log/nmos-service/` .

Installation Directories:

- Executable: `/usr/local/bin/nmos_service`
- Configuration: `/etc/nmos-service/config.jsonc` (fallback: `/etc/nmos-service/config.json`)

- Logs: `/var/log/nmos-service/`
- Cleanup script: `/usr/local/share/nmos-service/scripts/macos_cleanup.sh`

Service Management:

- Enable: `sudo launchctl load /Library/LaunchDaemons/tv.deltacast.nmos-service.plist`
- Disable: `sudo launchctl unload /Library/LaunchDaemons/tv.deltacast.nmos-service.plist`
- Start: `sudo launchctl start tv.deltacast.nmos-service`
- Stop: `sudo launchctl stop tv.deltacast.nmos-service`
- Status: `sudo launchctl list | grep tv.deltacast.nmos-service`

Uninstallation:

To completely remove the NMOS service from macOS:

```
sudo bash /usr/local/share/nmos-service/scripts/macos_cleanup.sh
```

This cleanup script will: - Stop and unload the service - Remove the service binary and configuration files - Remove log files - Clean up all installation directories

Linux Installation

The service is distributed in platform-specific formats: - **Debian/Ubuntu:** `nmos-service-linux.x64-<Version>.deb`

Installation Steps (Debian/Ubuntu):

1. Run the installer: `sudo apt install ./nmos-service-linux.x64-<Version>.deb`
2. Edit the configuration file located at `/etc/nmos-service/config.jsonc` . More details in the [Configuration](#) section.
3. The service is installed as a systemd unit and can be managed using `systemctl` commands.
4. Logs location are at `/var/log/nmos-service` .

Service Management (All Linux Distributions): - Start: `sudo systemctl start nmos-service` - Stop: `sudo systemctl stop nmos-service` - Restart: `sudo systemctl restart nmos-service` - Status: `sudo systemctl status nmos-service` - Enable auto-start: `sudo systemctl enable nmos-service` - Disable auto-start: `sudo systemctl disable nmos-service` - View logs: `sudo journalctl -u nmos-service -f`

Installation Directories: - Executable: `/usr/bin/` - Configuration: `/etc/nmos-service/` - Logs: `/var/log/nmos-service/` - Systemd service file: `/etc/systemd/system/nmos-service.service`

Configuration

The DELTACAST NMOS Service is configured through a JSON configuration file. This file controls all aspects of the service behavior, from network settings to logging verbosity.

Configuration File Location

- **Windows:** `C:\ProgramData\DELTACAST\nmos-service\config.jsonc`

- **macOS:** `/etc/nmos-service/config.jsonc`
- **Linux:** `/etc/nmos-service/config.jsonc`

Configuration File Structure

The configuration file is divided into logical sections:

```
{
  "nmos":
  {
    // NMOS-specific settings (primarily for out-of-board mode)
  },
  "service":
  {
    // Service operational settings
  },
  "nmos_on_board":
  [
    // On-board NMOS configuration per device
  ]
}
```

NMOS Configuration Section

Note: The `nmos` configuration section applies primarily to **out-of-board** deployment mode. For on-board devices, most settings are specified in the `nmos_on_board` array (see [On-Board Configuration](#) section). The exception is `uuid_seed`, which is shared across all deployment modes.

Basic Identity Settings

These settings define how the service identifies itself on the NMOS network:

```
"nmos": {
  "node_label": "DELTACAST NMOS Service"
,
  "node_description": "NMOS service for DELTACAST devices"
,
  "uuid_seed": "e44c4398-2cec-4e6c-87df-068e62a04644"
}
```

- **node_label** (string): Human-readable name for the NMOS Node. This appears in NMOS controllers and registries.
 - Default: `"DELTACAST NMOS Service"`
 - Example: `"Production Room A - DELTACAST Node"`
- **node_description** (string): Detailed description of the Node's purpose or location.
 - Default: `"NMOS service for DELTACAST devices"`
 - Example: `"Main production facility - 4K capture and playout"`
- **uuid_seed** (string): A UUID used to generate consistent UUIDs for resources across service restarts. Changing this value will cause the service to appear as a new Node in the registry. You must set this to a unique value for each deployment. It is generated

randomly at installation.

- Format: UUID v4 format

Network Configuration

These settings control how the service communicates on the network:

```
"nmos": {  
  "domain": "local."  
  ,  
  "node_api_port": 3212  
  ,  
  "connection_api_port": 3215  
  ,  
  "manifest_port": 3212  
  ,  
  "href_mode": "auto"  
  ,  
  "hostname": "my-machine"  
  ,  
  "host_ip": "192.168.0.92"  
}
```

- **domain** (string): DNS domain suffix for mDNS/DNS-SD discovery.
 - Default: "local." (for mDNS)
 - For unicast DNS-SD: Use your actual domain (e.g., "production.mycompany.com.")
- **node_api_port** (integer): TCP port for the IS-04 Node API.
 - Default: 3212
 - Must be available and not blocked by firewall
- **connection_api_port** (integer): TCP port for the IS-05 Connection Management API.
 - Default: 3215
 - Must be available and not blocked by firewall
- **manifest_port** (integer): TCP port for the IS-04 Manifest API.
 - Default: 3212
- **href_mode** (string): Controls whether host name/address fields use names, addresses, or both.
 - "auto" : automatic (nmos-cpp heuristic)
 - "name" : host name only
 - "addresses" : host addresses only
 - "both" : both host name and addresses
 - Default: "auto"
- **hostname** (string): Hostname of the machine running the service.
 - The service will attempt to auto-detect if not specified
 - Must be resolvable on your network
- **host_ip** (string): IP address to bind the NMOS APIs.
- **registry_address** (string): Force a specific NMOS registry hostname or IP address

instead of using DNS-SD discovery.

- Default: "" (empty, use DNS-SD discovery)
- Example: "10.10.3.40" or "registry.mynetwork.local"
- **registration_port** (integer): TCP port for the NMOS IS-04 Registration API when using a forced registry address.
 - Default: 3210
 - Only used when `registry_address` is specified

Security Configuration (HTTPS/TLS)

For production environments, you should enable HTTPS to secure communications:

```
"nmos": {  
  "client_secure": false  
,  
  "server_secure": false  
,  
  "ca_certificate_file": "/my/certs/ca.pem"  
,  
  "server_certificates":  
  [  
    {  
      "key_algorithm": "RSA"  
,  
      "private_key_file": "/my/servers/server-rsa-key.pem"  
,  
      "certificate_chain_file": "/my/servers/server-rsa-chain.pem"  
    }  
  ]  
}
```

- **client_secure** (boolean): Controls secure scheme publication and usage by nmos-cpp clients.
 - true : publish/use https and wss
 - false : publish/use http and ws
 - Default: if omitted, inherits `server_secure`
 - Recommended: keep equal to `server_secure` unless using a reverse-proxy topology
- **server_secure** (boolean): Enable HTTPS for all NMOS APIs.
 - Default: false
- **ca_certificate_file** (string): Path to the Certificate Authority (CA) certificate file.
 - Not used on Windows.
 - Required when `server_secure` is true
 - Used to verify client certificates and build trust chains
- **server_certificates** (array): List of server certificate configurations.
 - Not used on Windows.
 - **key_algorithm** : "RSA" or "ECDSA"
 - **private_key_file** : Path to the private key file (PEM format)
 - **certificate_chain_file** : Path to the certificate chain file (PEM format)

HTTPS Setup on Windows

On Windows, the service relies on the **Windows Certificate Store** for managing certificates and **HTTP.SYS** for the HTTP backend.

Note: On Windows, the `ca_certificate_file` and `server_certificates` configuration parameters are not used. All certificate management is handled through the Windows Certificate Store and HTTP.SYS. To enable HTTPS with consistent advertised URLs, set both `server_secure` and `client_secure` to `true`.

For a detailed step-by-step procedure for generating AMWA BCP-003-01 compliant certificates and configuring Windows SSL bindings, refer to [Appendix A: HTTPS Setup on Windows](#).

Disclaimer: The appendix procedure is provided for informational purposes only. DELTACAST does not guarantee the completeness or correctness of this procedure as certificate management and Windows security configuration are outside the scope of our core product support.

On-Board Configuration

The `nmos_on_board` array configures NMOS settings for on-board devices (DELTA-IP25-44-elp). Each entry matches devices by serial number and provides specific NMOS network settings.

```
"nmos_on_board": [  
  {  
    "serial_number": "SN123456"  
  ,  
    "hostname": "delta-board1"  
  ,  
    "host_ip": "10.10.3.41"  
  ,  
    "registry_address": ""  
  ,  
    "registration_port": 3210  
  ,  
    "domain": "local."  
  ,  
    "manifest_port": 3212  
  ,  
    "href_mode": "auto"  
  ,  
    "server_secure": false  
  ,  
    "ca_certificate_file": ""  
  ,  
    "server_certificates": [  
  ]  
  }  
]
```

On-Board Configuration Parameters

- `serial_number` (string): Serial number matching pattern for the device.
 - Matching behavior: Device S/N **contains** this string

- Example: "SN123456" matches device with S/N "SN123456789"
- Use empty string "" as catch-all/default for devices not matched by other entries
- **hostname** (string): Hostname for the on-board NMOS node.
 - This is the hostname advertised by the device on the network
 - Must be resolvable or use IP-based addressing
- **host_ip** (string): IP address for the on-board device's NMOS APIs.
 - This is the network interface IP where the device exposes NMOS endpoints
- **registry_address** (string): NMOS registry hostname or IP address for this device.
 - Default: "" (empty, use DNS-SD discovery)
 - Specify a registry address to force registration to a specific registry
- **registration_port** (integer): TCP port for the NMOS IS-04 Registration API.
 - Default: 3210
 - Used when `registry_address` is specified
- **domain** (string): DNS domain suffix for this device.
 - Default: "local."
- **manifest_port** (integer): TCP port for the IS-04 Manifest API.
 - Default: 3212
- **href_mode** (string): Controls URL formatting in NMOS resource hrefs.
 - Options: "auto", "name", "addresses", "both"
 - Default: "auto"
- **server_secure** (boolean): Enable HTTPS for this device's NMOS APIs.
 - Default: false

On-board secure mode behavior:

- In on-board mode, `client_secure` is intentionally not configurable.
- The embedded deployment does not expose reverse-proxy topology options.
- Therefore, secure client behavior is internally coupled to `server_secure`.
- If `server_secure=true`, on-board NMOS uses secure schemes consistently.
- **ca_certificate_file** (string): Path to CA certificate for HTTPS (non-Windows).
 - Default: "" (disabled)
- **server_certificates** (array): Server certificate configurations for HTTPS.
 - Default: [] (empty, disabled)
 - Same format as the main `nmos.server_certificates` array

Serial Number Matching Logic: - Entries are evaluated in order from top to bottom - First entry where device S/N contains the `serial_number` string is used - Use empty string "" as the last entry to provide default settings for unmatched devices

Service Configuration Section

Operational Settings

```
"service": {  
  "on_board": false  
,  
  "model_polling_interval_ms": 5000  
,  
  "verbosity": "debug"  
,  
  "max_log_files":  
5  
}
```

- **on_board** (boolean): Enable on-board NMOS support for DELTA-IP25-44-elp devices.
 - Default: `false`
 - When `true`, the service coordinates with the on-board NMOS implementation
 - Requires devices listed in the `nmos_on_board` configuration array
- **model_polling_interval_ms** (integer): How often (in milliseconds) the service polls DELTACAST boards for status updates.
 - Default: `1000` (1 second)
 - Minimum: `100` milliseconds
- **verbosity** (string): Logging level.
 - Options: `"fatal"`, `"error"`, `"warn"`, `"info"`, `"debug"`, `"trace"`
 - Default: `"info"`
- **max_log_files** (integer): Maximum number of log files to retain.
 - Default: `5`
 - Minimum: `1`
 - Older files are deleted when this limit is reached

Example Configurations

Example 1: Out-of-Board Configuration

This example shows a typical out-of-board deployment where the service runs on the host PC and manages DELTACAST PCIe boards:

```

{
  "nmos":
  {
    "node_label": "Production NMOS Node"
    ,
    "node_description": "NMOS service for DELTACAST devices"
    ,
    "uuid_seed": "e44c4398-2cec-4e6c-87df-068e62a04644"
    ,
    "domain": "local."
    ,
    "node_api_port": 3212
    ,
    "connection_api_port": 3215
    ,
    "registry_address": ""
    ,
    "registration_port": 3210
    ,
    "manifest_port": 3212
    ,
    "href_mode": "auto"
    ,
    "hostname": "prod-computer-01"
    ,
    "host_ip": "192.168.0.92"
    ,
    "client_secure": true
    ,
    "server_secure": true
    ,
    "ca_certificate_file": "/my/certs/ca.pem"
    ,
    "server_certificates":
    [
      {
        "key_algorithm": "RSA"
        ,
        "private_key_file": "/my/servers/server-rsa-key.pem"
        ,
        "certificate_chain_file": "/my/servers/server-rsa-chain.pem"
      }
    ]
  }
  ,
  "service":
  {
    "on_board": false
    ,
    "model_polling_interval_ms": 1000
    ,
    "verbosity": "info"
    ,
    "max_log_files": 50
  }
}

```

Example 2: On-Board Configuration

This example shows an on-board deployment for DELTA-IP25-44-elp devices where NMOS runs on the embedded device:

```

{
  "nmos":
  {
    "uuid_seed": "e44c4398-2cec-4e6c-87df-068e62a04644"
  }
,
  "service":
  {
    "on_board": true
  ,
    "model_polling_interval_ms": 1000
  ,
    "verbosity": "info"
  ,
    "max_log_files": 50
  }
,
  "nmos_on_board":
  [
    {
      "serial_number": "SN123456"
    ,
      "hostname": "delta-board1"
    ,
      "host_ip": "192.168.0.41"
    ,
      "registry_address": "192.168.0.40"
    ,
      "registration_port": 3210
    ,
      "domain": "local."
    ,
      "manifest_port": 3212
    ,
      "href_mode": "auto"
    ,
      "server_secure": false
    }
  ,
    {
      "serial_number": "SN576890"
    ,
      "hostname": "delta-default"
    ,
      "host_ip": "10.10.3.50"
    ,
      "registry_address": ""
    ,
      "registration_port": 3210
    ,
      "domain": "local."
    ,
      "manifest_port": 3212
    ,
      "href_mode": "auto"
    ,
      "server_secure": false
    }
  ]
}

```

Note: In on-board mode, only `uuid_seed` is used from the `nmos` section. All other NMOS

settings are specified per-device in the `nmos_on_board` array.

Applying Configuration Changes

After modifying the configuration file, you must restart the service for changes to take effect:

Configuration Validation

The service validates the configuration file at startup. If errors are detected:

- Error messages are written to the log file
- The service may fail to start or use default values
- Check the logs for detailed error messages

Common validation errors:

- Invalid JSON syntax (missing commas, brackets, quotes)
- Invalid port numbers (must be 1-65535)
- Invalid file paths for certificates
- Malformed UUIDs or IP addresses

Working with TX and RX Streams

Once the NMOS service is running and registered with your NMOS Registry, you can create and manage TX (transmit) and RX (receive) streams on your DELTACAST boards.

We assume you are familiar with the DELTACAST VideoMaster SDK and how to create TX and RX streams programmatically or via control applications.

Understanding TX Streams

TX streams represent media content being sent from your DELTACAST board to the network. Each TX stream consists of:

1. **Source:** The origin of the media (a video or audio input)
2. **Flow:** The formatted media stream with specific codec and timing parameters
3. **Sender:** The network endpoint that transmits the RTP stream

TX streams are created and controlled through the DELTACAST VideoMaster API or control applications. Once a TX stream is started, the NMOS service automatically:

- Creates the corresponding NMOS resources (Source, Flow, Sender)
- Registers them with the NMOS Registry
- Makes them discoverable by NMOS controllers
- Exposes IS-05 endpoints for connection management

Understanding RX Streams

RX streams represent media content being received from the network to your DELTACAST board. Each RX stream consists of:

1. **Receiver:** The network endpoint that receives and processes incoming RTP streams

RX streams are managed through NMOS IS-05 Connection Management. NMOS controllers can:

- Discover available Receivers
- Connect Receivers to remote Senders
- Configure transport parameters (multicast address, port, etc.)
- Monitor connection status

Opting Out of NMOS Exposure

Individual streams can be excluded from NMOS exposure by setting the `VHD_ST2110_SP_USE_NMOS` stream property to `false` in the stream configuration. When set to `false`, the stream is not registered with the NMOS registry and will not be visible to NMOS controllers, even if the service is running.

This is useful when a stream is used internally and should not be discoverable or controllable via NMOS.

Sample Applications

See VideoMaster SDK sample applications for examples of how to create and manage TX and RX streams programmatically. The NMOS service will automatically handle the NMOS resource mapping and registration for any streams created through the SDK.

Command-Line Options

The NMOS service binary supports the following command-line options:

Option	Description
<code>--version</code>	Print the service version and exit. Supported on all platforms.

Example:

```
nmos_service --version
```

This option is useful for verifying which version is installed before troubleshooting or upgrading.

Support

For technical support, questions, or issues with the DELTACAST NMOS Service, please contact **DELTACAST Support**:

- **Website:** <https://www.DELTACAST>

When contacting support, please have the following information ready:

- Operating system and version
- DELTACAST VideoMaster SDK version
- DELTACAST board model and driver version
- Service version
- Description of the issue

Licensing and Legal Information

Service License

This DELTACAST NMOS Service is distributed under the terms described in the `license.txt` file included with the distribution. Please review this file carefully to understand your rights and obligations.

Key Points:

- Review `license.txt` file for complete license terms and attribution requirements
- The `license.txt` file includes information about all Free and Open Source Software (FOSS) components used by this service
- Ensure compliance with all license terms when deploying the service
- Contact your vendor or legal counsel with licensing questions

Open Source Software Components

This service builds upon a rich ecosystem of Free and Open Source Software (FOSS) components. We are grateful to the open source community for their contributions. The following components are used either directly by this service or transitively through dependencies:

Core Dependencies

nmos-cpp (Sony)

- Purpose: Core NMOS implementation library
- License: Apache License 2.0
- Website: <https://github.com/sony/nmos-cpp>
- Used for: IS-04, IS-05, and other NMOS API implementations

nlohmann/json

- Version: 3.11.3
- License: MIT License
- Website: <https://github.com/nlohmann/json>
- Used for: JSON parsing and serialization

spdlog

- Version: 1.15.0
- License: MIT License

- Website: <https://github.com/gabime/spdlog>
- Used for: High-performance logging

Proprietary Components

DELTACAST VideoMaster SDK

- License: Proprietary license from DELTACAST
- Website: <https://www.DELTACAST>
- Used for: Hardware interface and control of DELTACAST boards
- **Note:** Separate license required from DELTACAST

Transitive Dependencies

The above components may include additional transitive dependencies. For a complete list of all dependencies and their license information, please refer to the `license.txt` file included with the distribution.

License Compliance

Responsibilities:

- Distributors and users must comply with all applicable licenses
- Attribution requirements must be maintained
- License files must be included in distributions
- Modified versions must indicate changes if required by license

Obtaining License Texts:

Full license texts for all components can be obtained from: - The `license.txt` file included with this distribution - The respective project websites listed above - The source code repositories

Warranty Disclaimer

This software is provided “as-is” without warranties of any kind, express or implied. See the `license.txt` file for complete warranty disclaimer and limitation of liability terms.

=====

Appendix A: HTTPS Setup on Windows

Disclaimer: This appendix provides a procedure for generating AMWA BCP-003-01 compliant certificates and configuring HTTPS on Windows for informational purposes only. This procedure is not officially supported by DELTACAST and is provided as-is without warranty. Certificate management and Windows security configuration are the responsibility of the system administrator and are outside the scope of DELTACAST product support.

Prerequisites

- Windows with Administrator privileges

- WSL (Windows Subsystem for Linux) installed for certificate generation
- OpenSSL (installed in WSL)

Step 1: Generate Certificates in WSL

Open your WSL terminal (e.g., Ubuntu) and execute the following commands:

1. Install OpenSSL and create the configuration file

```
sudo apt update && sudo apt install -y openssl
nano nmos-server.cnf
f
```

Paste the following configuration into `nmos-server.cnf`. **Important:** Replace `192.168.1.50` with your Windows machine's IP address and `MACHINE_HOSTNAME` with your actual hostname:

```
[ req ]
default_bits      = 256
default_keyfile    = server-key.pem
distinguished_name = req_distinguished_name
req_extensions     = v3_req
x509_extensions   = v3_ca
string_mask       = utf8only

[ req_distinguished_name ]
countryName       = B
stateOrProvinceName = Lieg
localityName      = Lieg
organizationName  = Deltacast.t
commonName        = MACHINE_HOSTNAME

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage         = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
subjectAltName   = @alt_names

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints     = critical, CA:TRUE
keyUsage              = critical, digitalSignature, cRLSign, keyCertSig

[ alt_names ]
DNS.1 = localhost
IP.1  = 127.0.0.1
IP.2  = 192.168.1.50
DNS.2 = MACHINE_HOSTNAME
```

Save and exit (Ctrl+O, Enter, then Ctrl+X).

2. Generate keys and build the PFX container

```
# Generate the Root CA
openssl ecparam -name prime256v1 -genkey -noout -out ca-key.pem
openssl req -new -x509 -sha256 -days 3650 -key ca-key.pem -out ca-cert.pem \
-subj "/CN=NMOS-Root-CA" -config nmos-server.cnf

# Generate the NMOS Server Certificate
openssl ecparam -name prime256v1 -genkey -noout -out server-key.pem
openssl req -new -key server-key.pem -out server.csr \
-subj "/CN=nmos-cpp-server" -config nmos-server.cnf
f
openssl x509 -req -in server.csr -CA ca-cert.pem -CAkey ca-key.pem \
-CAcreateserial -out server-cert.pem -days 825 -sha256 \
-extfile nmos-server.cnf -extensions v3_req

# Export to PFX format (you will be prompted for a password, e.g., "nmos")
openssl pkcs12 -export -out server.pfx -inkey server-key.pem \
-in server-cert.pem -certfile ca-cert.pem
```

3. Transfer files to Windows

Copy `ca-cert.pem` and `server.pfx` from WSL to a Windows location (e.g., `C:\nmos-cpp\certs\`).

Step 2: Import Certificates into Windows Certificate Store

Open **PowerShell as Administrator** and run:

1. Install the Root CA certificate

```
Import-Certificate -FilePath "C:\nmos-cpp\certs\ca-cert.pem" `
-CertStoreLocation Cert:\LocalMachine\Root
```

2. Install the Server PFX certificate

```
# Replace the password with the one you set during PFX export
$password = ConvertTo-SecureString "nmos" -AsPlainText -Force
Import-PfxCertificate -FilePath "C:\nmos-cpp\certs\server.pfx" `
-CertStoreLocation Cert:\LocalMachine\My `
-Password $password -KeyStorageFlags PersistKeySe
t
```

Step 3: Configure HTTP.SYS SSL Binding

Still in **Administrator PowerShell**:

1. Retrieve the certificate thumbprint

```
$cert = Get-ChildItem Cert:\LocalMachine\My | Where-Object { $_.Subject -match "nmos-cpp-server"
}
$cert.Thumbprint
```

Copy the displayed thumbprint (e.g., EF1C0F9B9F6AC7308A6B109A0FB08160B1385433).

2. Create the SSL binding using netsh

Replace `certhash` with your actual thumbprint and `ipport` with your NMOS API port (e.g., 3212 for IS-04):

```
netsh http add sslcert ipport=0.0.0.0:3212 certhash=EF1C0F9B9F6AC7308A6B109A0FB08160B1385433 app
id={00112233-4455-6677-8899-AABBCCDDEEFF}
```

Repeat for other NMOS API ports (e.g., 3215 for IS-05) using the same certificate hash.

Windows should return: "SSL Certificate successfully added."

Step 4: Enable HTTPS in Configuration

Edit your `config.jsonc` and set:

```
"nmos": {
  "server_secure": true
}
```

Restart the DELTACAST NMOS Service for changes to take effect.