NDI® Discovery Service

VERSION 6.2.0

Introduction

In modern networked media workflows, efficient discovery and management of audio and video sources are essential. The **NDI Discovery Service** is a dedicated server and protocol designed to facilitate seamless search, monitoring, and control of **NDI receivers** and indexing of respective **sources** within a network. This can run as a service (on a remote server) or an application (on your local machine) that accepts incoming connections with senders, finders, and receivers, and coordinates amongst them all to ensure they are all visible to each other. By providing a centralized registry, the service enables devices to automatically announce their presence, retrieve real-time status updates, and establish low-latency connections with minimal configuration. In NDI version 6.2, this new **NDI Discovery Service** adds new capabilities of discovery, monitor and control for **NDI receivers**, compared to previous versions, that only supported advertising and listing **NDI sources**.

To enable seamless integration with third-party applications, a **C API** is available for integration, allowing developers to communicate directly with the **NDI Discovery Service**. This API provides functionalities for querying available **NDI** sources, subscribing to real-time updates, and managing connections efficiently, ensuring interoperability across various applications and workflows.

As a final note, the NDI Discovery is an alternative to using **mDNS**, which allow a quite significant reduction in network traffic compared to the latter.

NDI Discovery Server

The **Server** will be a **standalone application** or as a **system service**, named **NDI Discovery Service**, and will provide flexibility for different deployment needs (platforms).

Both 32-bit and 64-bit versions of the **NDI Discovery Service** are available, although the 64-bit version is recommended. The server will use very little CPU usage although, when there are a very large number of connections (representing sources, receiver listeners and receiver advertisers), it might require RAM and some network traffic between all sources to coordinate source lists.

Configuration

To configure the discovery server for NDI clients, you can use **NDI Access Manager** (included in the NDI Tools bundle) to enter the IP address of the machine running the discovery server.

Alternatively, you can run the discovery server with a command-line option to specify the network interface (NIC) it should use:

```
"NDI Discovery Service.exe" -bind 192.168.1.100
```

This ensures that the discovery server advertises only on the specified IP address. Similarly, you can specify a port number for the discovery server using:

```
"NDI Discovery Service.exe" -port 5400
```

This enables you to use a non-default port or run multiple discovery servers for different groups of sources on the same machine. If a port number of 0 is specified, the operating system will automatically assign a port, which will be displayed at runtime.

Command-Line Options

"NDI Discovery Service.exe" -bind <ip_address> -port <port_number> -config <json path> -help

- **General Options (Windows & Linux):**
  - `-bind <ip_address>` - This is an optional argument that specifies the IP address to bind the service to (default: 0.0.0.0).
  - `-port <port_number>` - This is an optional argument that sets the port number (default: 5959).
  - `-config <json path>` - This is an optional argument that loads configuration from a JSON file. This option will override both -bind and -port options as it will take precedence
  - `-help` - Displays available command-line options.
- **Additional options for Linux:**
  - `-service` - Runs the NDI Discovery in **service mode** for background execution.
- **JSON configuration file:**

  The default paths for the configuration file are:

  - **Linux**:
    - `/etc/ndi/ndi-discovery-service.v1.json`
    - `/usr/local/etc/ndi/ndi-discovery-service.v1.json`
  - **Windows**:

    `C:\ProgramData\NDI\ndi-discovery-service.v1.json`

  An example of the JSON configuration file can be seen below:

```
{
  "binding": "127.0.0.1",
  "port_no": "5959"
}
```

Console Output

When executed as a standalone application, the NDI Discovery will provide a console output with some relevant information regarding the number of registered sources, listeners and advertisers, that connect or disconnect at a given time. This console will not show up when the application is running in service mode

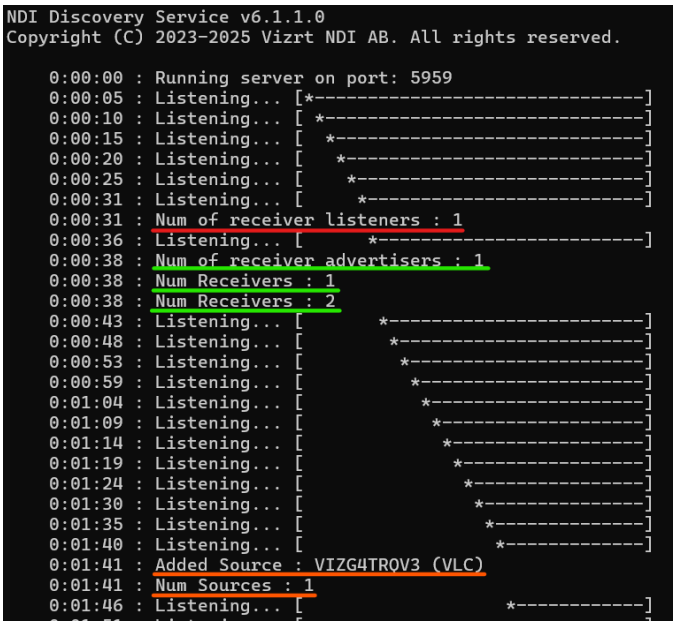


Figure 1-Console log example

Log explanation:

- Underlined in red, is the notification when a new receiver listener connects.
- Underlined in green, is the notification when a new receiver advertiser connects and the respective NDI receivers it advertises.
- Underlined in orange, a notification of a newly advertised NDI source.
- The “Listening…” is just a progress to signal that the service is working properly, waiting for new connections, either from receiver or source clients.

NDI Discovery Clients

Clients are entities that communicate with the NDI Discovery server. They can be categorized as follows:

Source Clients

Source clients should be configured to connect with the discovery server instead of using mDNS to locate sources. When there is a discovery server, the SDK will use both mDNS and the discovery server for *finding and receiving* to locate sources on the local network that are not on machines configured to use discovery. In the future, Sources will also have the capability to be monitored and controlled by the NDI Discovery protocol

For Source *senders*, if an **NDI Discovery** server is specified, the mDNS mechanism will not be used; these sources will *only* be visible to other finders and receivers that are configured to use the **NDI Discovery** server.

- **Source Senders**: These clients advertise or register NDI sources with the server, making them easily discoverable.

- **Source Finders**: These clients query the server to locate sources that have been advertised by any Source Sender.

Redundancy and Multiple Servers

Within NDI version 6 there is full support for redundant NDI discovery servers. When one configures a discovery server it is possible to specify a comma delimited list of servers (e.g., “192.168.10.10, 192.168.10.12”) and then they will all be used simultaneously. If one of these servers then goes down then as long as one remains active then all sources will always remain visible; as long as at least one server remains active then no matter what the others do then all sources can be seen.

This multiple server capability can also be used to ensure entirely separate servers to allow sources to be broken into separate groups which can serve many workflow or security needs.

Receiver Clients

Receiver clients are a new addition to **NDI Discovery**, beginning from NDI version 6.2, as earlier versions only supported source discovery. In addition to discovery, these new clients can also be used to monitor and control remote NDI receivers.

- **Receiver Advertisers**
    - Advertise NDI receivers
- **Receiver Listeners**
    - Monitor and Control NDI receivers
        - Monitoring is related to getting fleeting or more static information from a given receiver. In Version 1, we currently support events for the following receiver properties:
            - source-name: **NDI Source** to which the receiver is connected to
            - source-url: The URL of the **NDI Source**
            - connection-state: Connected or Disconnected
            - audio-present : True of False
            - audio-channels: Number of audio channels
            - audio-sample-rate: Audio sample rate
            - audio-receive-mode
            - video-present: True or False
            - video-codec: Video codec name (i.e. shq2)
            - video-resolution: i.e. 1920x800
            - video-frame-rate: i.e. 24000/1001
            - video-frame-type: i.e. progressive
            - video-color-primaries: i.e. bt_709

        - video-transfer-function: i.e. bt_709
        - video-matrix-coefficients: i.e. bt_709
        - video-has-alpha: True or False
        - video-receive-mode: i.e. single_tcp
    - In the future the number of properties to be monitored will be expanded

Receiver clients only connect to the Discovery Server directly, and unlike source clients, they rely solely on the server to advertise and to listen/query for receivers in the network. The new APIs can be consulted in the NDI SDK documentation for more details.

Redundancy and Multiple Servers

- **Receiver Advertisers:** Like their source counterpart, Receiver Advertisers can also accept a comma-separated list of servers (e.g., "192.168.10.10,192.168.10.12"). In this case, the receivers will be announced to all specified servers simultaneously.
- **Receiver Listeners:** The current implementation relies on the user to instantiate the listener and implement logic that enables the Receiver Listener to detect when the server is no longer operational. Once detected, it should then connect to an alternative, redundant Discovery Service server available on the same network.

Installer Availability and Platform Support

The Discovery Server (DS) installer is available for both **Windows** and **Linux**, ensuring broad compatibility across different environments.

We provide pre-built DS installer binaries for the following architectures:

- **Intel x86 / x86-64** (commonly used in desktops, servers, and cloud instances)
- **ARM** (suitable for embedded systems and energy-efficient computing)
- **Raspberry Pi** (optimized for low-power SBCs)

Windows Installer Features

The **Windows installer** for the Discovery Server (DS) provides flexible installation options to suit different deployment needs.

- Silent Installation
    - The installer supports **silent installation** via command-line arguments, enabling automated deployments without user interaction.
    - Example usage:

      ```
      setup.exe /verysilent /port=1234 /binding=127.0.0.1
      ```

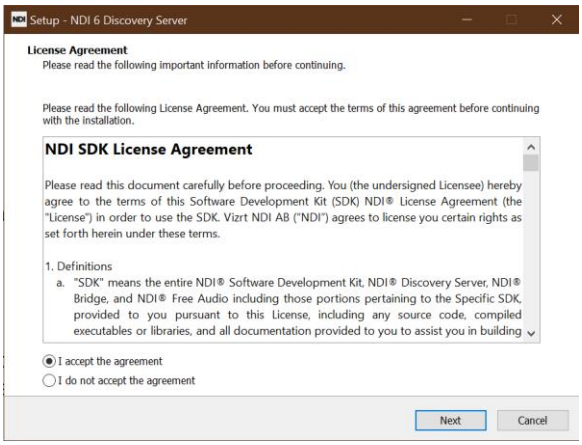
        - **/verysilent** installs **Discovery Service** without displaying any UI prompts.

- **/port=<port number>** specifies the listening port.
- **/binding=<ip address>** restricts DS to listen only on the local machine.

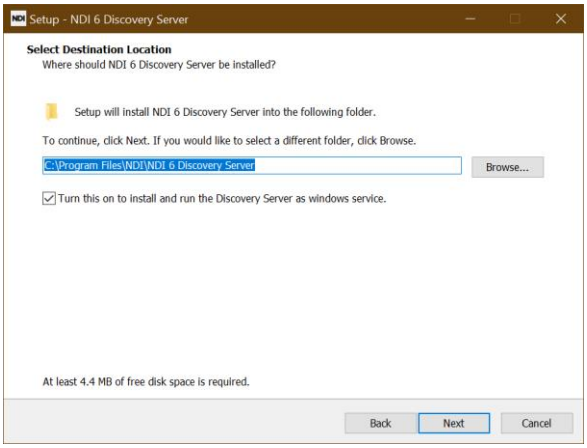
- Interactive Installation

  After starting the installer, you'll be greeted with the following:

  - License agreement



  - Installation location selection



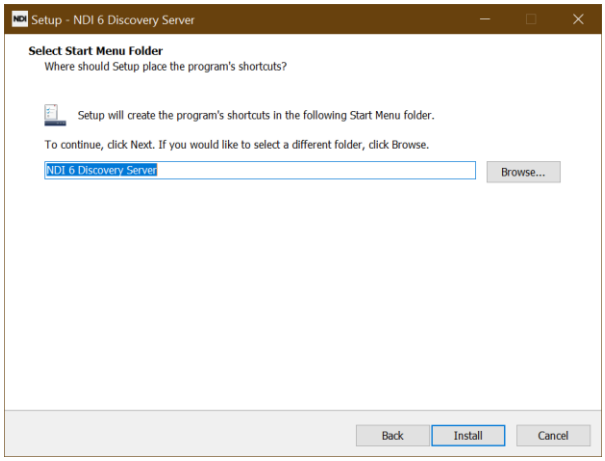
    If the user wants NDI Discovery to run as a service, they must select the corresponding checkbox. Otherwise, it will be installed as a standalone application that can be launched manually like any other program.

    It's also possible, through the application itself, to register it later as a service by running the following commands:

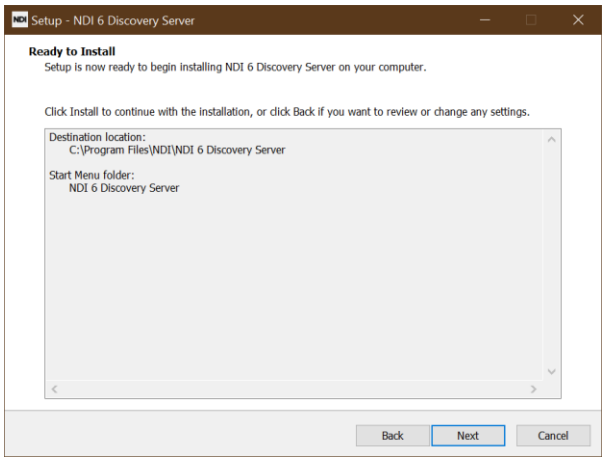
    - `"NDI Discovery Service.exe" install` - Installs the service. (or in this case re-installs it)
    - `"NDI Discovery Service.exe" remove` - Uninstalls the service. (from the command line)

- *"NDI Discovery Service.exe" start* - Starts the service. (from the command line)
- *"NDI Discovery Service.exe" stop* - Stops the service. (from the command line)

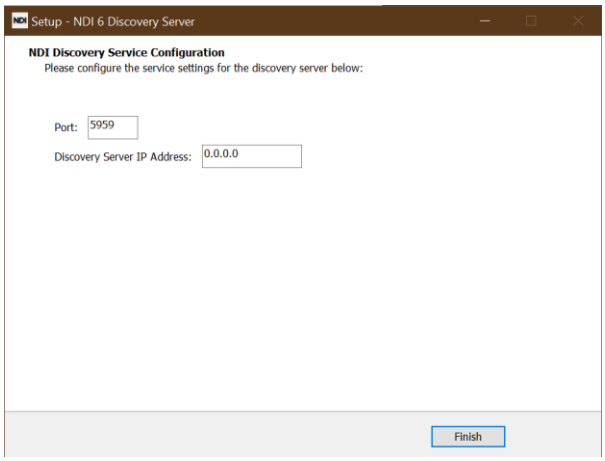
- Change folder destination location



- Install screen



- Port selection and address binding selection



This screen allows the user to configure the port where the service will be launched and the IP address to which it will be bound (useful if the machine has multiple network interfaces). If left unchanged, the address will be selected automatically, and the service will bind to the first available network device and IP detected on the system.

- At the end of the installation, this documentation will be shown

Linux Installer Features

There's also a Linux installer available that will universally install binaries for all supported platforms, being those:

- aarch64-newtek-linux-gnu
- aarch64-rpi4-linux-gnueabi
- arm-newtek-linux-gnueabihf
- arm-rpi1-linux-gnueabihf
- arm-rpi2-linux-gnueabihf
- arm-rpi3-linux-gnueabihf
- arm-rpi4-linux-gnueabihf
- i686-linux-gnu
- x86_64-linux-gnu

The installer is an automated installation script named **"Install_NDI_Discovery_Server_v6.sh"**. Upon execution, the user will be presented with a license agreement document that must be accepted for the installation to proceed automatically. The installation process is straightforward, with all binaries being placed in the **Install_NDI_Discovery_Server_v6/bin** directory, located within the folder where the installation script was initially executed.

Running as a Service in Linux

SystemD framework

1. To run NDI Discovery as a service on Linux using **systemd**, the user must use the provided template, fill it out appropriately, and save it as **/etc/systemd/system/ndi-discovery.service**.

```
[Unit]
```

```
Description=NDI Discovery Service
After=network.target

[Service]
Type=simple
EnvironmentFile=-/etc/default/ndi-discovery
WorkingDirectory=${INSTALL_PATH}
ExecStart=${INSTALL_PATH}/ndi-discovery-server -service
Restart=always
RestartSec=5
User=nobody
Group=nobody

[Install]
WantedBy=multi-user.target
```

2. It then needs to replace the ${INSTALL_PATH} with the actual path of the binary to be launched.
3. To enable and start the service, the user needs to run the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable ndi-discovery
sudo systemctl start ndi-discovery
```

4. This setup allows users to simply edit the **/etc/default/ndi-discovery** file to specify their installation directory, making it flexible without modifying the service file itself.

Init.d (for systems without systemd)

1. Create the following script and save it at **/etc/init.d/ndi-discovery**

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          ndi-discovery
# Required-Start:    $network
# Required-Stop:     $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: NDI Discovery Service
# Description:       Starts the NDI Discovery service with the -service argument.
### END INIT INFO

# Load configuration file if it exists
CONFIG_FILE="/etc/default/ndi-discovery"

if [ -f "$CONFIG_FILE" ]; then
```

```
    . "$CONFIG_FILE"
fi

# Default path if not set in /etc/default/ndi-discovery
INSTALL_PATH=${INSTALL_PATH:-"/opt/ndi-discovery"}

DAEMON="$INSTALL_PATH/ndi-discovery-server"
ARGS="-service"
PIDFILE="/var/run/ndi-discovery.pid"
LOGFILE="/var/log/ndi-discovery.log"

start() {
    echo "Starting NDI Discovery..."
    if [ -f "$PIDFILE" ]; then
        echo "NDI Discovery is already running."
        exit 1
    fi
    cd "$INSTALL_PATH" || exit 1
    nohup "$DAEMON" $ARGS > "$LOGFILE" 2>&1 &
    echo $! > "$PIDFILE"
    echo "NDI Discovery started."
}

stop() {
    echo "Stopping NDI Discovery..."
    if [ -f "$PIDFILE" ]; then
        kill "$(cat $PIDFILE)" && rm -f "$PIDFILE"
        echo "NDI Discovery stopped."
    else
        echo "NDI Discovery is not running."
    fi
}

restart() {
    stop
    sleep 2
    start
}

status() {
    if [ -f "$PIDFILE" ]; then
        echo "NDI Discovery is running (PID: $(cat $PIDFILE))."
    else
        echo "NDI Discovery is not running."
    fi
```

```
}

case "$1" in
    start) start ;;
    stop) stop ;;
    restart) restart ;;
    status) status ;;
    *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```

2. Make it executable with:

   sudo chmod +x /etc/init.d/ndi-discovery

3. Change INSTALL_PATH to the path where the binary is located.
4. Register the service with:

   sudo update-rc.d ndi-discovery defaults

5. Start the service

   sudo service ndi-discovery start

6. Enable the service to start on boot

   sudo update-rc.d ndi-discovery enable