mediARC™

# Runtime Environment

Administrator Manual
MAA0001

# Table of Contents

# Preface

# 1 Preface

## 1.1 Summary

This is an Administrator Manual describing the general mediARC system architecture, mediARC's core modules, native client applications and NOA products compatible with mediARC. In order to provide deeper insight in mediARC's most basic functionalities, primary setup and runtime administration of its core modules is provided.

## 1.2 Target Audience

This Administrator Manual targets mediARC system administrators and system architects.

## 1.3 Prerequisites

The following skills are required to follow the instructions shown in this document:

- basic knowledge of mediARC
- knowledge of local IT infrastructure
- proficiency in server administration and networking

The following tools are required to follow the instructions shown in this document:

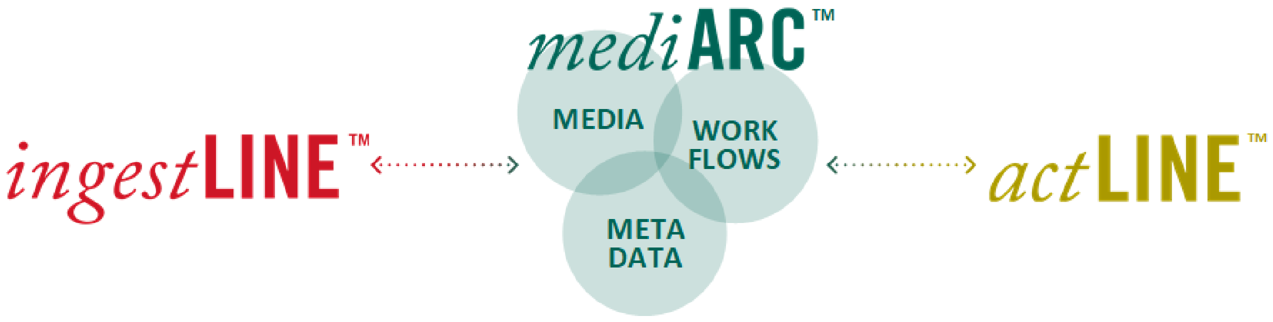
- access to mediARC servers
- administrative user rights on mediARC servers
- administrative rights to local network resources
- mediARC installers

# What Is mediARC

# 2 What Is mediARC

Broadcasters and archivists today face daunting challenges when it comes to ensuring that media content is preserved and accessible for the long term. Many broadcasters see their media archive as the heart and soul of their organization. Yet the sheer number of content sources and formats, and the fact that content is constantly changing and evolving, makes it difficult to manage, especially when it comes to digitizing, describing, linking, and storing that content in a way that makes it easy to find and access. Broadcasters and archivists need an asset management system that can ingest, describe, archive, and deliver content so that it fits archival requirements not just for the present, but also for decades into the future. Whereas production departments typically focus only on the next playout date, archival description of content has to serve history, cultural identity, and cultural heritage.

Based on NOA's proven, easy-to-use, open technologies, the mediARC system is a flexible framework for Archive Asset Management that makes the whole process easier and more efficient for broadcasters and archives of all sizes. mediARC's modular structure means you can scale the application perfectly to fit your own performance, security, and distributed usage requirements.



mediARC answers the need for structured and flexible content management that has to last for decades, and comes with a range of powerful tools that can digest petabytes of information. mediARC can handle the complex structures needed for proper description and efficient use of an archive's content, so you can link media data to metadata and storage on a granular level, and even apply FRBR (Functional Requirements of Bibliographic Records). A powerful and exceptionally fast search engine, capable of combining structured and full-text search capabilities, allows easy access to information through a graphical user interface.

Developed in cooperation with dedicated broadcasters, the mediARC system is designed to meet the requirements in three fundamental areas:

**Metadata:**

- Descriptive data of both general and media-specific nature.
- Annotation of content for semantic cataloging.
- Extensive relationships of information.

**Media:**

- Content that is stored in the system's media archive as essence data.
- Additional technical information such as Markers.

**Workflows:**

- Pre-configured processes managing any transaction in the system like ingest or retrieval of archive content.
- Automatic processes and human interaction.

*mediARC is a flexible framework for Archive Asset Management. It allows for tightly integrated handling of both media content and its associated metadata, whilst putting interaction with the system under the control of a dedicated workflow engine. The system is highly suitable for the needs of broadcasting corporations with high information turnover, as well as for large audio archives with a complex metadata structure.*

# System Architecture Overview

# 3 System Architecture Overview

mediARC is a client-server computer system consisting of various modules and applications connected via a computer network.

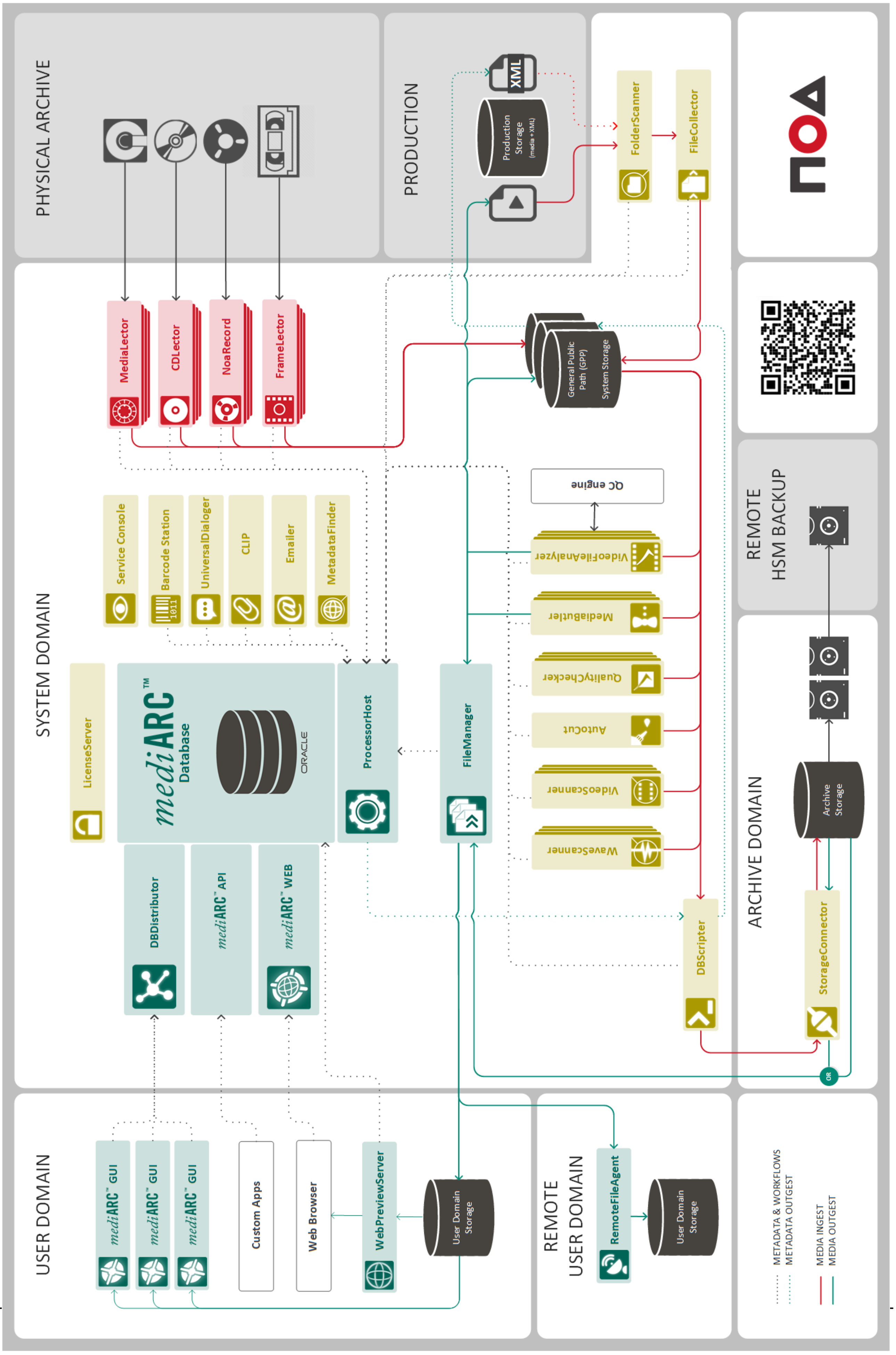
Internally, the system is logically organized into 3 types of so called Access Domains (separated on the network level for security):

- System Domain: contains NOA technology that is responsible for media and metadata processing, ingesting, and outgesting. It is the bridge between users and the archive. Typically ingests and outgests have needs for transformations such as rewrapping, transcoding or the need to deliver different sets of metadata depending on the target beneficiary. System Domain is the place where a central temporary storage (also called General Public Path or GPP) in the form of simply registered n- network resources is exclusively used to do the job.
- Archive Domain: contains the archive. It may be resident inside an HSM or a disk storage. Depending on the kind of archive storage, the archive domain is always accessed via a dedicated method, the most secure method is given via the NOA Storage Connector. An institution may have one or many archive domains.
- User Domain: contains the users of the archive who are allowed to see and hear content via proxies and get access to a copy of archive files. User Domains may be separate local networks (such as organizational entities) or remote sites (such as in regional stations). An institution may have one or many user domains.

*For further information about mediARC Access Domains, please refer to the mediARC Domains TechNote.*

The components of the mediARC system can be divided into 4 categories:

- mediARC modules (green)
- mediARC client applications (green)
- actLINE products (yellow)
- ingestLINE products (red)


PHYSICAL ARCHIVE
PRODUCTION
XML
Production Storage (media + XML)
FolderScanner
FileCollector
SYSTEM DOMAIN
MediaLector
CDLector
NoaRecord
FrameLector
General Public Path (GPP)
System Storage
Service Console
Barcode Station
UniversalDialoger
CLIP
Emailer
MetadataFinder
LicenseServer
mediARC™ Database
ORACLE
ProcessorHost
FileManager
QC engine
VideoFileAnalyzer
MediaBugler
QualityChecker
AutoCut
VideoScanner
WaveScanner
DBDistributor
mediARC™ API
mediARC™ WEB
DBScripter
ARCHIVE DOMAIN
Archive Storage
StorageConnector
OR
REMOTE HSM BACKUP
USER DOMAIN
mediARC™ GUI
mediARC™ GUI
mediARC™ GUI
Custom Apps
Web Browser
WebPreviewServer
User Domain Storage
REMOTE USER DOMAIN
RemoteFileAgent
User Domain Storage
METADATA & WORKFLOWS
METADATA OUTGEST
MEDIA INGEST
MEDIA OUTGEST

The following mediARC modules represent the core parts of the mediARC system:

| | |
|---|---|
| mediARC Database | The heart of every mediARC system, built on market-leading Oracle technology. |
| DBDistributor | Enables managed database connections of the mediARC GUI. |
| ProcessorHost | Connects Processors to the mediARC database. |
| FileManager | Responsible for file transfers, both within the archive, and in and out of the archive. |
| RemoteFileAgent | Remotely receives files from FileManager and synchronizes users from Windows Active Directory. |
| WebPreviewServer | Streaming server for prelisten/preview requests from mediARC WEB. |
| mediARC API | PL/SQL package for API access to the mediARC system. |

Normally, every mediARC installation also includes the following actLINE apps:

| | |
|---|---|
| LicenseServer | Allows to count and control the amount of granted licenses. |
| StorageConnector | A file access agent, which handles the access of archive storage to the rest of the system. |
| ServiceConsole | Provides a centralized, real-time overview of nearly all NOA processors running on distributed servers. |

## 3.1 mediARC Database

### 3.1.1 Overview

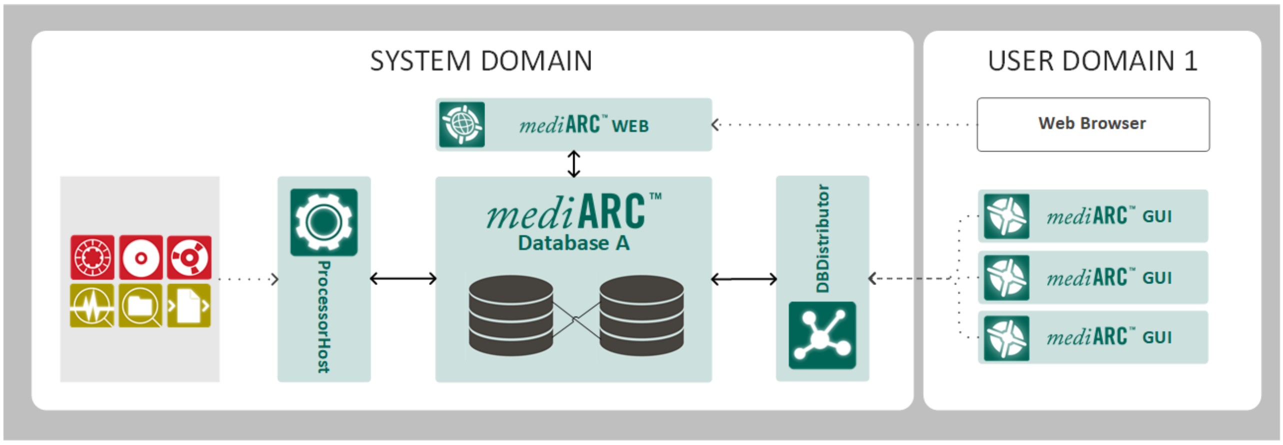
The mediARC Database is a database schema which is generally deployed under ORACLE technology by NOA engineers during a system installation. It is either created by dedicated scripts or by the import of a database dump.

The mediARC database is typically installed on a separate Linux server or server cluster dedicated for this role.

*Warning: It is highly recommended that customers do not manipulate with the mediARC Database directly and always rely on NOA professional services instead!*

For the sake of security and optimum performance, direct connections to the mediARC Database are limited to special mediARC modules:

- DBDistributor
- ProcessorHost
- mediARC WEB
- mediARC API

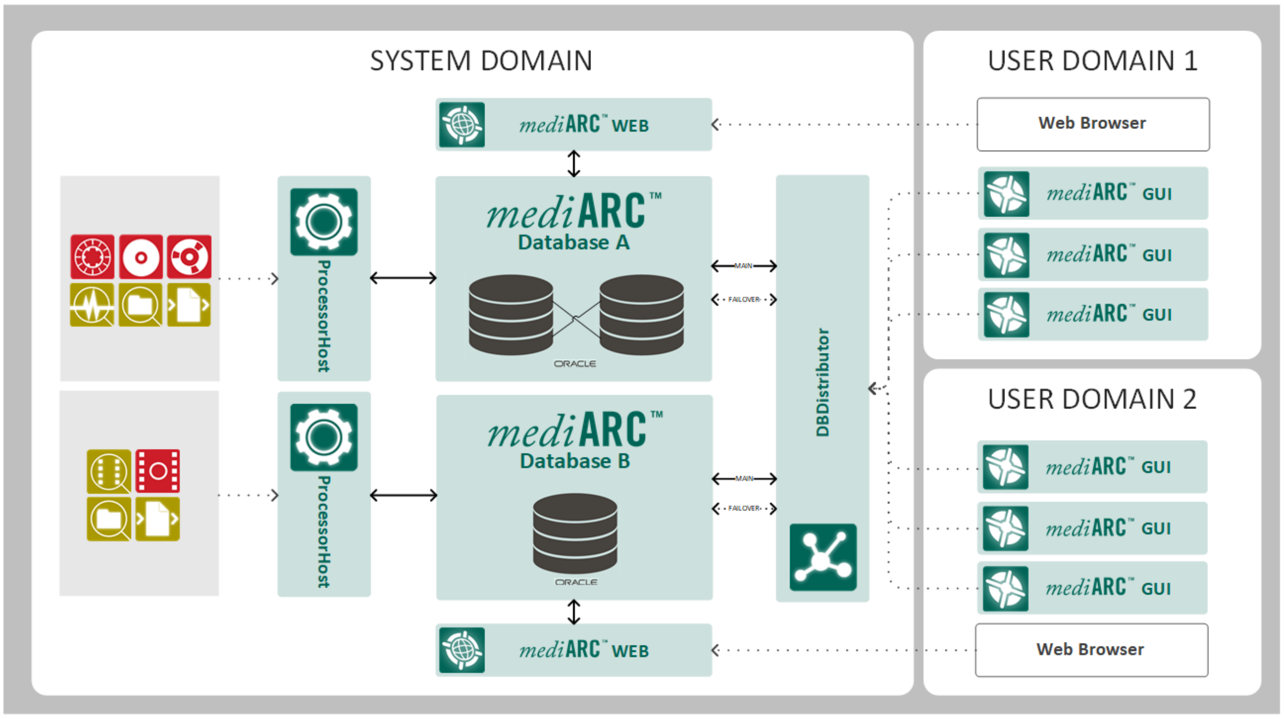


*A common mediARC Database setup*

Even though more complex database setups are possible with mediARC, these are far less common. System complexity rises with:

- the use of more than one Access Domain of one type (System Domain, Archive Domain, User Domain)
- the use of more than one mediARC Database

*Note that system complexity does not rise with the use of various high-availability (HA) setups for the mediARC database, as from mediARC's point of view these are being handled by and within the Oracle platform itself and not by NOA software.*



*An example of a very complex mediARC database architecture.*

*For different scenarios, please see chapter DBDistributor > Failover Database Connection.*

### 3.1.2 Installation

- Install Oracle Server
  o Installing Oracle Server is quite easy following the setup instructions.

- Create an instance
  o "Unicode standard UTF-8 AL32UTF8" as character set
  o "Data Warehouse" as database type

*Note that as the Oracle instance could be created at a later stage, it is possible (and recommended) to create the Starter Database and then use the Advanced option to configure it correctly.*

- Install Oracle Client
  o Follow setup instructions and choose "Administrator" installation type when asked.

*Note that DBDistributor needs a 32bit version of the Oracle Client to run properly!*

- Configure Client connection
  - Using Net Configuration Assistant provided by Oracle is quite easy. Just select "Local Net service Name configuration" and choose what you want to do with a connection:
    - Add
    - Reconfigure
    - Delete
    - Rename
    - Test
  - Important things to know when making a connection:
    - Service Name = Global Database Name defined while creating the Oracle Instance
    - Host name = Oracle server name or IP address

*Note that it is recommended only to change the other settings (like protocol and port) if it is absolutely necessary!*
*Please always test the connection!!!*

- Create a mediARC schema
  - An empty mediARC database can be created from existing scripts.
  - With existing installations however, a migration from the old database(s) is usually possible and the Customer may receive an Oracle dump (export file).

*Never try to manipulate with the mediARC database without consulting with NOA first!*

### 3.1.3 Maintenance

Generally there are no special requirements for maintenance. Normal Oracle rules apply.

*Note that while it is highly recommended to keep the patch level of the mediARC database up-to-date, these updates are not enforced or done automatically by NOA and are always first discussed and agreed upon with the Customer.*

#### 3.1.3.1 Patching The mediARC Schema

Patches are provided for updating an existing mediARC schema under Service & Maintenance Contract in order to make latest features available and to fix broken functionality.

*Note that DB patches are applied by NOA engineers. It is, however, possible to agree on a different scenario based on the Customer's wishes and resources.*

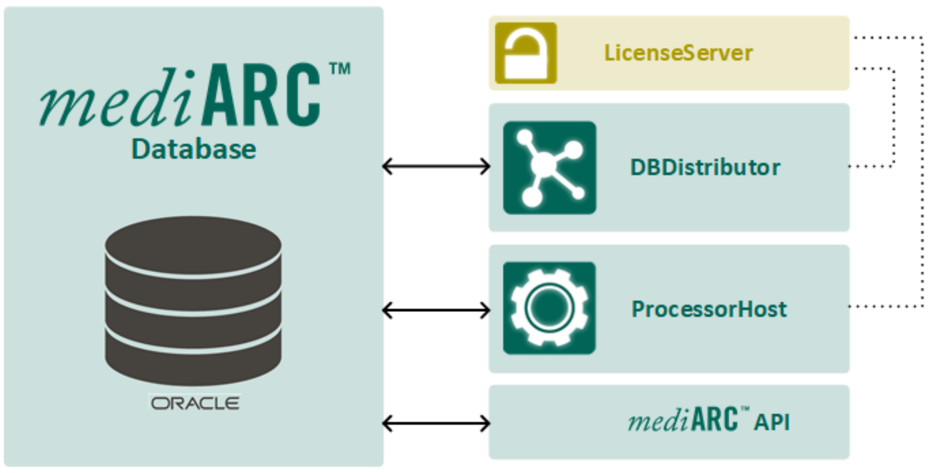
As it is not recommended for Customers to manipulate with the mediARC database directly, this chapter will give only a brief general overview of how mediARC database patching works:

- a new DB patch is released by NOA
- a JIRA ticket is opened by NOA, informing the Customer of a new available DB patch for mediARC
- if no reasons not to proceed are identified, a patch window is scheduled (normally 0.5 - 1 hours)
- before the DB patch is applied:
  - o the mediARC system is shut down (see chapter mediARC Startup and Shutdown Sequence) and the availability of an up-to-date database backup is checked
  - o if not already available, a database management tool (e.g. Oracle SQL Developer (SQLD)) is installed and configured on a machine inside the Customer's network (DBD or PH server)
  - o DB patching scripts are applied by NOA
  - o error logs are checked for errors
  - o any non-ignorable errors are resolved by NOA Customer Support
  - o once all DB patches were executed correctly, the system can be started (with new software versions where necessary)

## 3.2 mediARC Core Modules

At the heart of each mediARC system is the mediARC database. In order to use the database, several core modules are present to facilitate the connections of users and other system modules and applications. These include:

- DB Distributor
- ProcessorHost
- LicenseServer
- mediARC API



*mediARC Core Modules*

## 3.3 mediARC Network Ports

The following are the default network ports used by mediARC and its components:

| Application | Description | Port |
|---|---|---|
| StorageConnector | Client Interface | 5707 |
| | Service Console | 5505 (reserved, N/A) |
| FileCollector | Service Console | 5511 |
| FolderScanner | Service Console | 5512 |
| WaveScanner | Service Console | 5513 |
| DBScripter | Service Console | 5517 |
| LicenseServer | Client Interface | 5557 |
| | Old Legacy Client Interface (jobDB Client only) | 5756 |
| | Service Console | 5557 |
| DBDistributor | Client Interface | 5767 |
| | Service Console | 5567 |
| WaveButler | Service Console | 5577 |
| MediaButler | Service Console | 5579 |
| ProcessorHost | Client Interface | 5787 |
| | Web Interface (Flow Logs, Pico UI) | 5788 |
| | Service Console | 5587 |
| RemoteFileAgent | Client Interface | 5791 |
| | Service Console | 5591 |
| EMailer | Service Console | 5593 |
| Dactylo Indexer | Service Console | 5595 |
| FileManager | Service Console | 5597 |
| VideoFileAnalyzer | Service Console | 5514 |
| VideoScanner | Service Console | 5515 |
| WebPrelistenAgent (obsolete) | Service Console | 5599 |
| WebPreviewServer | Client Interface (HTTP) | 5799 |

## 3.4 mediARC Startup and Shutdown Sequence

By design mediARC is a hierarchical system of interconnected modules and applications, governed by licenses.
Due to this fact, a system startup and/or shutdown needs to be carried out in special order.

**Startup**

| Order | Application, Service |
|---|---|
| 1 | Oracle database |
| 2 | LicenseServer |
| 3 | ProcessorHost, DBDistributor |
| 4 | FileManager, DBScripter, WaveButler, MediaButler |
| 5 | the rest of the modules and applications |

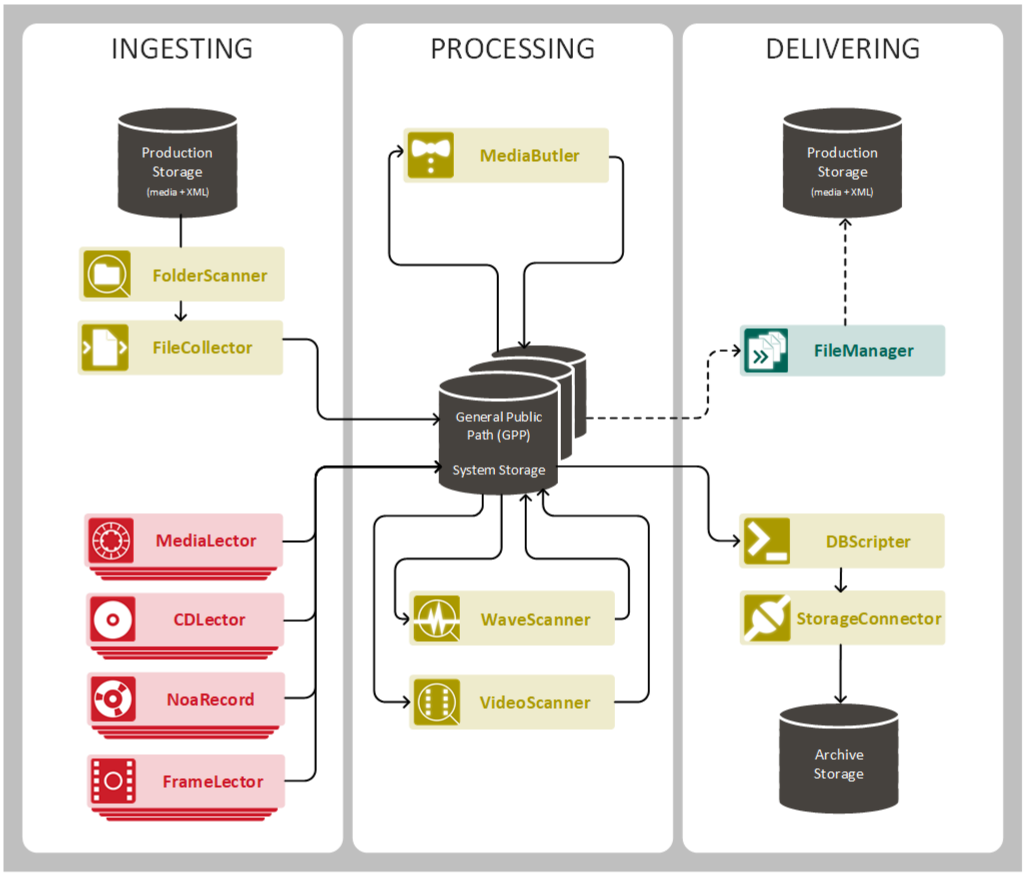
**Shutdown**

| Order | Application, Service |
|---|---|
| 1 | NOA Record, CD Lector, MediaLector, FrameLector, AutoCut, BarcodeStation, CLIP, DBScripter, EMailer, FileCollector, FolderScanner, MetaDataFinder, QualityChecker, ServiceConsole, UniversalDialoguer, VideoFileAnalyzer, VideoScanner, WaveScanner |
| 2 | FileManager, DBScripter, WaveButler, MediaButler |
| 3 | ProcessorHost, DBDistributor |
| 4 | LicenseServer |
| 5 | Oracle database |

*Note that RemoteFileAgent and StorageConnector do not have a LicenseServer connection, which is why they can be turned on and off anytime during the procedure.*

## 3.5 General Public Path

The General Public Path (often referred to as GPP) is the storage path for inter-task file communication during the execution of a Flow Template. It resides on a centrally reachable physical storage, to which only Processors have read and write access.



*Being mediARC's central storage, the GPP can serve as destination, workplace, and source.*

The GPP acts mostly as intermediate and temporal cache-path to serve transcoding engines or 3rd party processes such as CLIP. n- paths are registered within mediARC and are known to the ProcessorHost. An intelligent round robin distribution is handled by the ProcessorHost which provides disk resources dynamically based upon free space. An amount of up to 20-30 GPP's are quite typical (and not limited) so that additional GPP's can quite easily be added to the pool. Supported environments for GPPs are CIFS/SMB shares only.

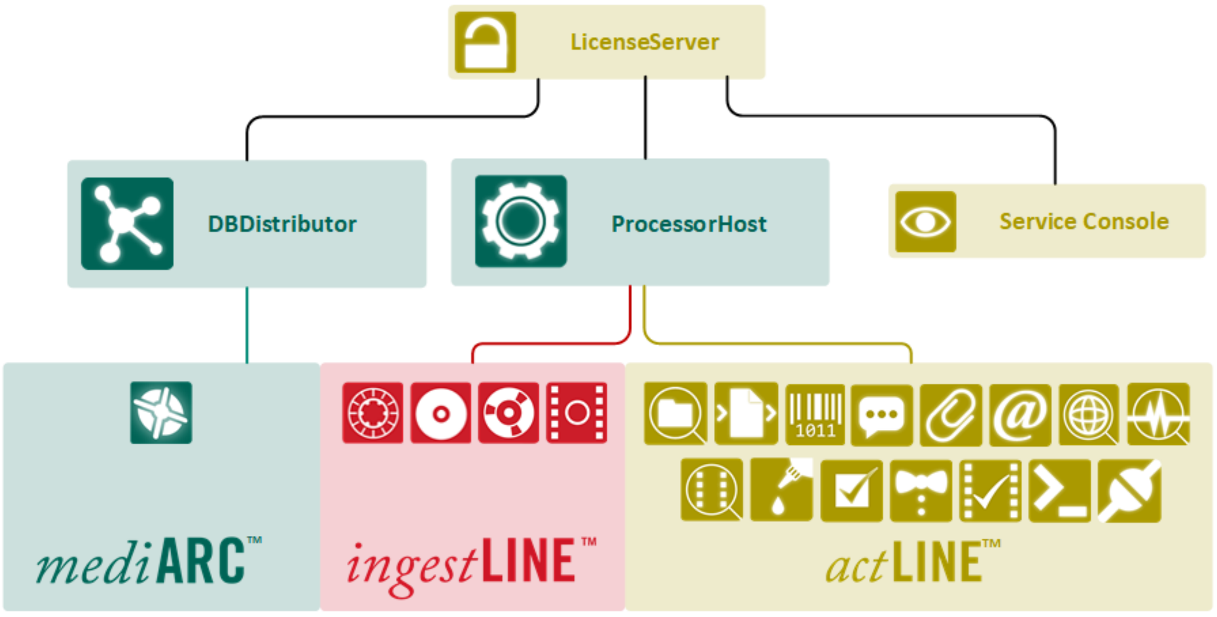
*For further information, please refer to the mediARC Domains TechNote.*

# LicenseServer (LS)

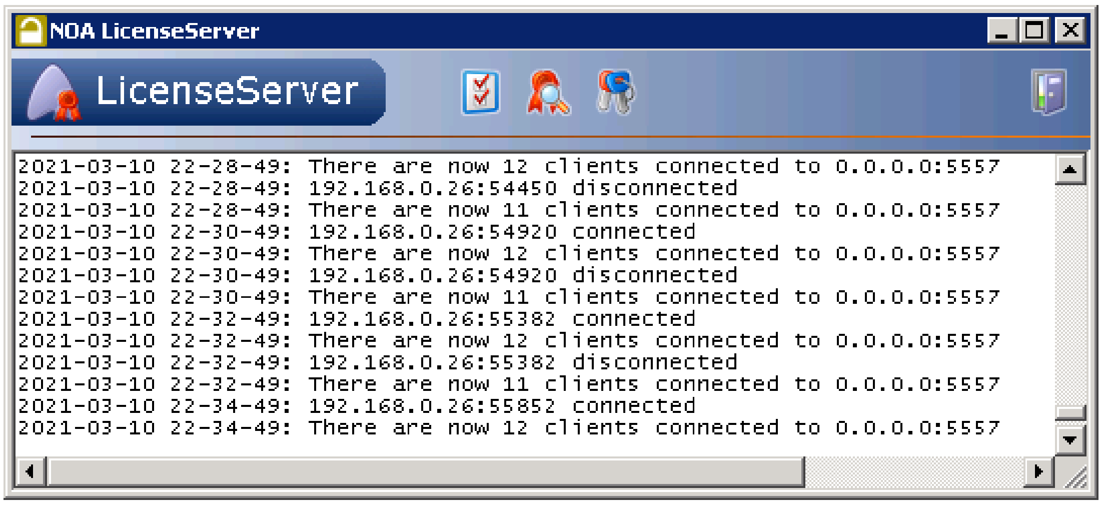
# 4 LicenseServer (LS)

## 4.1 Overview

The LicenseServer in a mediARC system takes the role to define the data used, to count and control the amount of granted licenses and to specify the user rights.



In a mediARC environment the LicenseServer needs to run on a separate Windows server, usually in combination with the DBDistributor and the ProcessorHost.



```
2021-03-10 22-28-49: There are now 12 clients connected to 0.0.0.0:5557
2021-03-10 22-28-49: 192.168.0.26:54450 disconnected
2021-03-10 22-28-49: There are now 11 clients connected to 0.0.0.0:5557
2021-03-10 22-30-49: 192.168.0.26:54920 connected
2021-03-10 22-30-49: There are now 12 clients connected to 0.0.0.0:5557
2021-03-10 22-30-49: 192.168.0.26:54920 disconnected
2021-03-10 22-30-49: There are now 11 clients connected to 0.0.0.0:5557
2021-03-10 22-32-49: 192.168.0.26:55382 connected
2021-03-10 22-32-49: There are now 12 clients connected to 0.0.0.0:5557
2021-03-10 22-32-49: 192.168.0.26:55382 disconnected
2021-03-10 22-32-49: There are now 11 clients connected to 0.0.0.0:5557
2021-03-10 22-34-49: 192.168.0.26:55852 connected
2021-03-10 22-34-49: There are now 12 clients connected to 0.0.0.0:5557
```

*The LicenseServer Window*

In the main window you can view the license information, configure the LicenseServer and User Rights.

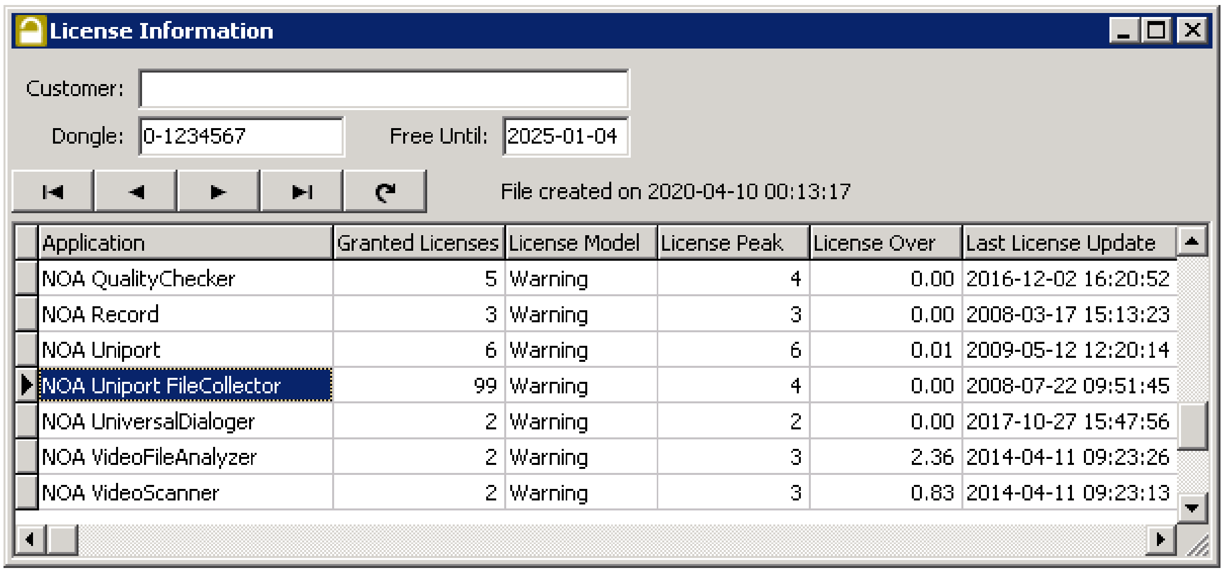
The following buttons are available:

Change Ini-Data: shows the application wide settings window.

Show License Information: shows the license file contents.

Change User Rights: shows the 'Edit users, groups and rights' window used for managing the user rights for the ServiceConsole.

*Note that under normal conditions, LicenseServer will be executed as a Windows service. On occasions, it is possible to run LicenseServer as an executable program. Please make sure to stop the service before running the executable.*


License Information

Customer:

Dongle: 0-1234567    Free Until: 2025-01-04

File created on 2020-04-10 00:13:17

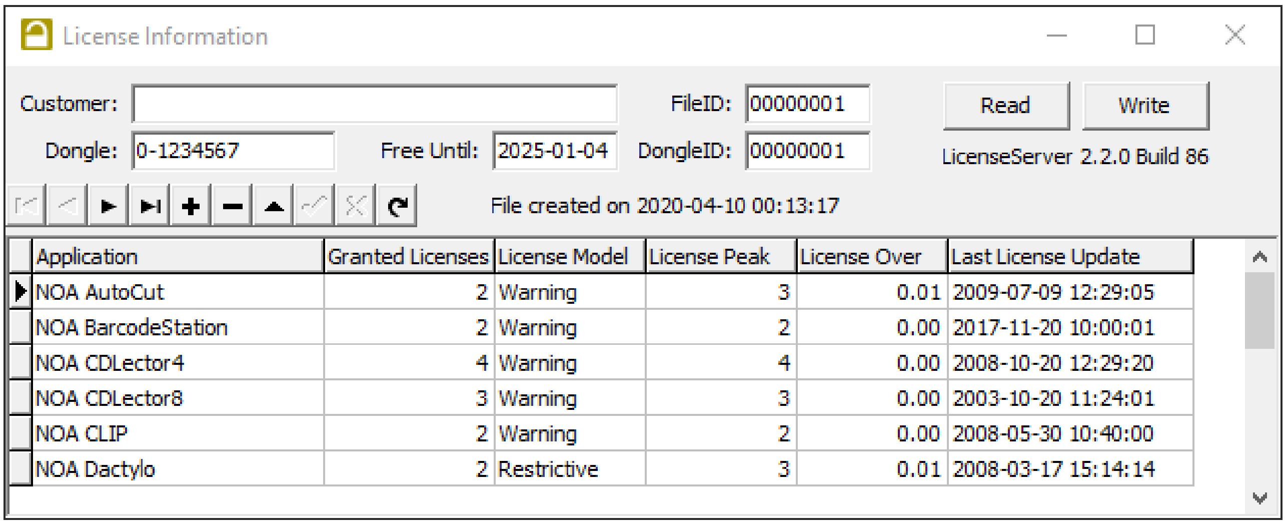
| Application | Granted Licenses | License Model | License Peak | License Over | Last License Update |
|---|---|---|---|---|---|
| NOA QualityChecker | 5 | Warning | 4 | 0.00 | 2016-12-02 16:20:52 |
| NOA Record | 3 | Warning | 3 | 0.00 | 2008-03-17 15:13:23 |
| NOA Uniport | 6 | Warning | 6 | 0.01 | 2009-05-12 12:20:14 |
| NOA Uniport FileCollector | 99 | Warning | 4 | 0.00 | 2008-07-22 09:51:45 |
| NOA UniversalDialoger | 2 | Warning | 2 | 0.00 | 2017-10-27 15:47:56 |
| NOA VideoFileAnalyzer | 2 | Warning | 3 | 2.36 | 2014-04-11 09:23:26 |
| NOA VideoScanner | 2 | Warning | 3 | 0.83 | 2014-04-11 09:23:13 |

*The License Information window*

*Warning: Stopping the LicenseServer causes all related programs to stop working! Please refer to the mediARC Startup and Shutdown Sequence chapter to plan this step accordingly...*

## 4.1.1 License File

*Note that the LicenseServer is locked against a license file. The "license.dat" file is an 128-bit encrypted binary file which contains the "granted licenses". Every NOA customer will obtain this file from support@noa-archive.com with its contents depending from the purchased licenses for the "Applications". The license file may be further locked against a date or a WIBUBOX device. The WIBUBOX may be either attached locally by USB or Parallel Port, or provided remotely through a TCP/IP connection.*



Customer:
FileID: 00000001
Dongle: 0-1234567
Free Until: 2025-01-04
DongleID: 00000001
LicenseServer 2.2.0 Build 86
File created on 2020-04-10 00:13:17

| Application | Granted Licenses | License Model | License Peak | License Over | Last License Update |
|---|---|---|---|---|---|
| NOA AutoCut | 2 | Warning | 3 | 0.01 | 2009-07-09 12:29:05 |
| NOA BarcodeStation | 2 | Warning | 2 | 0.00 | 2017-11-20 10:00:01 |
| NOA CDLector4 | 4 | Warning | 4 | 0.00 | 2008-10-20 12:29:20 |
| NOA CDLector8 | 3 | Warning | 3 | 0.00 | 2003-10-20 11:24:01 |
| NOA CLIP | 2 | Warning | 2 | 0.00 | 2008-05-30 10:40:00 |
| NOA Dactylo | 2 | Restrictive | 3 | 0.01 | 2008-03-17 15:14:14 |

*License File Contents*

The license file contains information about the granted licenses:

- Customer: the name of the customer.
- FileID: the license file identification number.
- Dongle: the serial number of the WIBU key dongle.
- Free Until: the date until the LicenseServer can be used without a dongle.
- DongleID: the last read dongle identification number
- File created on: the date the license file has been created.
- Granted licenses:
  - o Application: the name of the licensed application.
  - o Granted Licenses: the number of purchased licenses.
  - o License Model: Restrictive vs. Warning. See License Models.
  - o License Peak: the number of exceeded licenses. See License Models.
  - o License Over: the hours of exceeded number of licenses.
  - o Last License Update: the date and time of the last change.

## 4.1.2 License Models

A NOA customer can be granted two different kinds of licenses.

- Restrictive: restricts the number of available licenses to exactly the amount granted and specified in the license file.
- Warning: allows the customer to use more licenses than the amount granted and specified in the license file, however, with a notification message during application startup, that a certain amount of granted licenses has been exceeded and that this will be logged. Also, the LicenseServer will log the count of exceeded licenses (License Peak) and the count of hours the number of granted licenses had been exceeded (License Over).

*NOA applies the Warning model to customers who might first only see how many licenses are really needed. Typically, after a specific time the amount of exceeded licenses* *and the corresponding amount of hours are known and the customer is notified to purchase the correct amount of licenses. During this procedure also the license model is changed by NOA towards Restrictive.*

### 4.1.3 License Terms and Conditions

Current License Terms and Conditions can always be found in the dedicated section of NOA's official website www.noa-archive.com.

## 4.2 Installation

To install or reinstall the LicenseServer, simply run the installer and follow the instructions.
During installation you can decide whether you want to install LicenseServer as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

**Prerequisites**

- LicenseServer needs a pre-installed OracleClient (32-bit version) pointing to the right database schema.

*Note that in 64-bit versions of Windows, the LicenseServer needs to be installed in a directory whose full path does not contain special characters (e.g. C:\NOA\) and not in the default directory "C:\Program Files (x86)\NOA\" because the brackets "(" and ")" will break the connection!*

*Make sure that you place the "license.dat" file in the same directory as the licenseserver.exe application!*

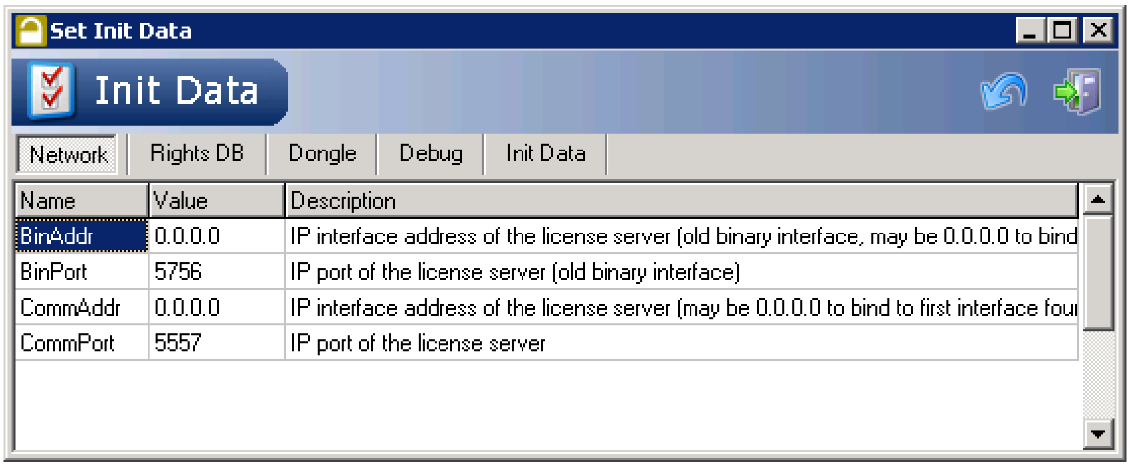
## 4.3 Configuration

*The configuration data of the LicenseServer is stored in the Windows registry, thus it is available for the service as well as for the application. Please take care which Windows user account you use when configuring the application as the settings will be available only for this user account. This is important when running the service version later on because the service must then be configured to use the same user account which was used during the application configuration step.*

*The registry entries can be found in HKEY_USERS\.DEFAULT\Software\NOA\LicenseServer.*

## 4.3.1 Init Data Window

There are some application-wide settings that can be found either using the Change Ini Data button .



| Name | Value | Description |
|---|---|---|
| BinAddr | 0.0.0.0 | IP interface address of the license server (old binary interface, may be 0.0.0.0 to bind |
| BinPort | 5756 | IP port of the license server (old binary interface) |
| CommAddr | 0.0.0.0 | IP interface address of the license server (may be 0.0.0.0 to bind to first interface fou |
| CommPort | 5557 | IP port of the license server |

*The Set Init Data window of the LicenseServer*

The following Init Data settings are used by the LicenseServer:

**Network Tab**
BinAddress: The old binary (as opposed to the new NoaComm) IP interface address of the license server (may be 0.0.0.0 to bind to first interface found). Defaults to "0.0.0.0".
BinPort: The old binary (as opposed to the new NoaComm) IP interface port number of the license server. Defaults to "5756".
CommAddress: The IP interface address of the license server (may be 0.0.0.0 to bind to first interface found). Defaults to "0.0.0.0".
CommPort: The IP interface port number of the license server. Defaults to "5557".

**Rights DB Tab**
NTServer: The Hostname or IP address of the domain server providing user/group information. If empty, the local machine will be used.
ExplizitUsers: The Username tokens (comma separated values) which can be used for authentication toward applications (in ServiceConsole for example) which are independent of actual domain users. The access permission of the user can then be defined in User Rights Management. See User Rights.

*Note that the string "explizit" as used in the Init Data window contains a spelling error. Unfortunately fixing it would break backwards compatibility with previous versions of LicenseServer.*

**Dongle Tab**
Dongle Server: The IP address of the WIBU dongle server (if the dongle is not connected to the local machine).

**Debug Tab**
DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\LSlogs\LS.log"). If a file name is provided, debug messages will be written to this file. The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

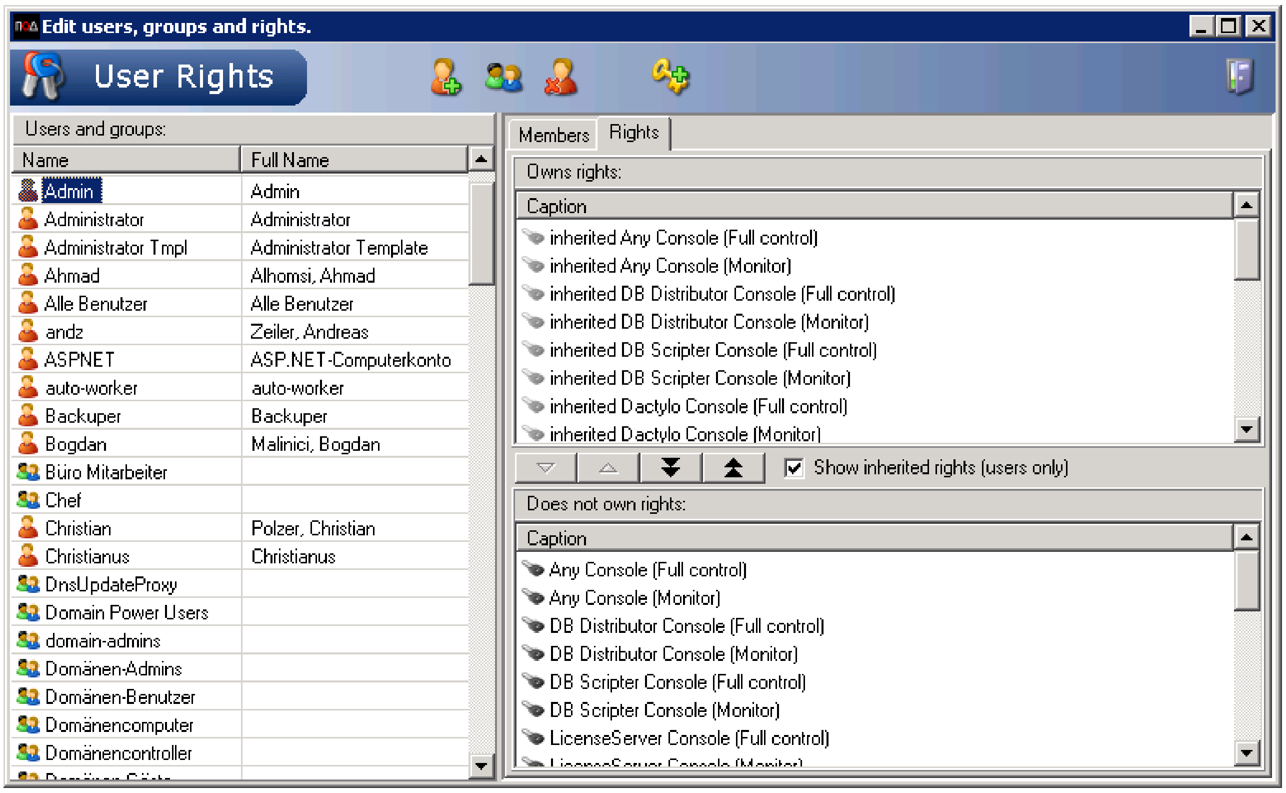
**Init Data Tab**
Settings for the jobDB Database, not applicable to mediARC systems.

### 4.3.2 User Rights Window

Whenever mediARC administrators need to connect to one of the mediARC modules running as Windows service with the ServiceConsole, setting appropriate user rights becomes necessary. Only users who have granted rights may be able to see NOA services within the ServiceConsole. This User Rights Management (URM) is defined by the LicenseServer, in the User Rights window.

There are two ways to create or modify existing user rights:
- If the LicenseServer is running as a GUI application, click the Change User Rights button.
- If the LicenseServer is running as a Windows service, stop the LicenseServer and use the RightsConfig tool (available as NoaRightsConfig.exe in the LicenseServer installation directory).

Both ways open the same window.


Edit users, groups and rights.

User Rights

Users and groups:

| Name | Full Name |
|---|---|
| Admin | Admin |
| Administrator | Administrator |
| Administrator Tmpl | Administrator Template |
| Ahmad | Alhomsi, Ahmad |
| Alle Benutzer | Alle Benutzer |
| andz | Zeiler, Andreas |
| ASPNET | ASP.NET-Computerkonto |
| auto-worker | auto-worker |
| Backuper | Backuper |
| Bogdan | Malinici, Bogdan |
| Büro Mitarbeiter | |
| Chef | |
| Christian | Polzer, Christian |
| Christianus | Christianus |
| DnsUpdateProxy | |
| Domain Power Users | |
| domain-admins | |
| Domänen-Admins | |
| Domänen-Benutzer | |
| Domänencomputer | |
| Domänencontroller | |

Members | Rights

Owns rights:

| Caption |
|---|
| inherited Any Console (Full control) |
| inherited Any Console (Monitor) |
| inherited DB Distributor Console (Full control) |
| inherited DB Distributor Console (Monitor) |
| inherited DB Scripter Console (Full control) |
| inherited DB Scripter Console (Monitor) |
| inherited Dactylo Console (Full control) |
| inherited Dactylo Console (Monitor) |

☑ Show inherited rights (users only)

Does not own rights:

| Caption |
|---|
| Any Console (Full control) |
| Any Console (Monitor) |
| DB Distributor Console (Full control) |
| DB Distributor Console (Monitor) |
| DB Scripter Console (Full control) |
| DB Scripter Console (Monitor) |
| LicenseServer Console (Full control) |

*The LicenseServer's User Rights Window*

In the User Rights window, a mediARC administrator is able to define which Windows user/group is allowed to access which NOA service application.

The following buttons are available:

Create New User: creates a new User in the list.

Create New Group: creates a new User Group in the list.

Delete Selected User or Group: deletes the currently selected User or Group.

Load Rights From File: allows to load a rights file (see next chapter) with user rights definitions.

The displayed list of User accounts and Groups is either created locally or retrieved from the domain server which defined in the Rights DB tab of the LicenseServer's Init Data window.

*Note that if you want to select another domain controller from which the Windows user/group accounts are retrieved you have to change the NTServer setting in the Init Data window of the LicenseServer and restart the LicenseServer.*

User rights are granted to the currently selected User or Group by moving them from the "Does not own rights" pool to the "Owns rights" list in the Rights tab:

- Single arrow: moves the selected user right.
- Double arrow: moves all user rights.

*In order for the changes to be saved, LicenseServer needs to be restarted. The LicenseServer keeps the User Rights DB stored in the Windows registry to load settings on startup and save them on server shutdown.*

All Users with granted user rights are being stored in the Windows registry HKEY_LOCAL_MACHINE\Software\NOA\UserRights\Users
All Groups with granted user rights are being stored in the Windows registry HKEY_LOCAL_MACHINE\Software\NOA\UserRights\Groups

### Explicit Users

From time to time it is necessary to grant access even to users who are not member of the Windows domain the LicenseServer is running in. This is possible via the ExplizitUsers setting in the Init Data window of the LicenseServer, where authentication strings can be added manually (e.g. '*donaldduck*'). To add more than one authentication string, the entries have to be separated with a comma ','.

The manually added authentication strings will show up in the User/Group list of the User Rights window, as a User of the (virtual) group ExplizitUsers. It is possible to set the access rights for these "explicit users" the same way as for the regular Windows users.

*The authentication strings of explicit users can be used in manually added registrations in the ServiceConsole (see the ServiceConsole chapter).*

## 4.3.2.1 User Rights Definitions

*The LicenseServer keeps a User Rights DB stored in the Windows registry to load settings on startup and save them on server shutdown. The default registry path to the User Rights DB is HKEY_LOCAL_MACHINE\Software\NOA\UserRights on the server the LicenseServer is running on.*

In order to define or add a set of user rights to the User Rights DB, either a new rights file needs to be created or an already existing rights file need to be taken and loaded from the User Rights window.

To create a new rights file, use the following format:
• one right per line
• right name and description separated by a comma ','

A valid rights file may look like this:

```
LICENSESERVER.MONITOR, LicenseServer Console (Monitor)
LICENSESERVER.CONTROL, LicenseServer Console (Full control)
```

```
DBDISTRIBUTOR.MONITOR, DBDistributor Console (Monitor)
DBDISTRIBUTOR.CONTROL, DBDistributor Console (Full control)
DBSCRIPTER.MONITOR, DBScripter Console (Monitor)
DBSCRIPTER.CONTROL, DBScripter Console (Full control)
```

The name represents the Windows registry key for the user right.
The description represents the display name used in the User Rights window.

User rights can be set for the all applications, supported by the ServiceConsole. Each application has two different types of access rights:

- .MONITOR: The connected user has the ability to view service status but cannot change service configuration.
- .CONTROL: The connected user has the ability to view service status and to change service configuration (i.e. full control)

The following User Rights definitions are supported:

- BarcodeStation
  - o ServiceConsole.BarcodeStation.Monitor
  - o ServiceConsole.BarcodeStation.Control
- Clip
  - o CLIP.Monitor
  - o CLIP.Control
- DBDistributor
  - o DbDistributor.Monitor
  - o DbDistributor.Control
- DBScripter
  - o ServiceConsole.DbScripter.Monitor
  - o ServiceConsole.DbScripter.Control
- Emailer
  - o ServiceConsole.EMailer.Monitor
  - o ServiceConsole.EMailer.Control
- FileManager
  - o ServiceConsole.FileManager.Monitor
  - o ServiceConsole.FileManager.Control
- LicenseServer
  - o LicenseServer.Monitor
  - o LicenseServer.Control
- MediaButler
  - o ServiceConsole.MediaButler.Monitor
  - o ServiceConsole.MediaButler.Control
- MetaDataFinder
  - o MetaData.Monitor
  - o MetaData.Control
- ProcessorHost
  - o ServiceConsole.ProcessorHost.Monitor
  - o ServiceConsole.ProcessorHost.Control

- RemoteFileAgent
  - RemoteFileAgent.Monitor
  - RemoteFileAgent.Control
- Uniport
  - Uniport.Monitor
  - Uniport.Control
- Uniport FileCollector
  - Uniport.Monitor
  - Uniport.Control
- Uniport FolderScanner
  - Uniport.Monitor
  - Uniport.Control
- VideoFileAnalyzer
  - ServiceConsole.VideoFileAnalyzer.Monitor
  - ServiceConsole.VideoFileAnalyzer.Control
- VideoScanner
  - ServiceConsole.VideoScanner.Monitor
  - ServiceConsole.VideoScanner.Control

*Universal user rights, independent from the application, can be given by the ANY user right:*

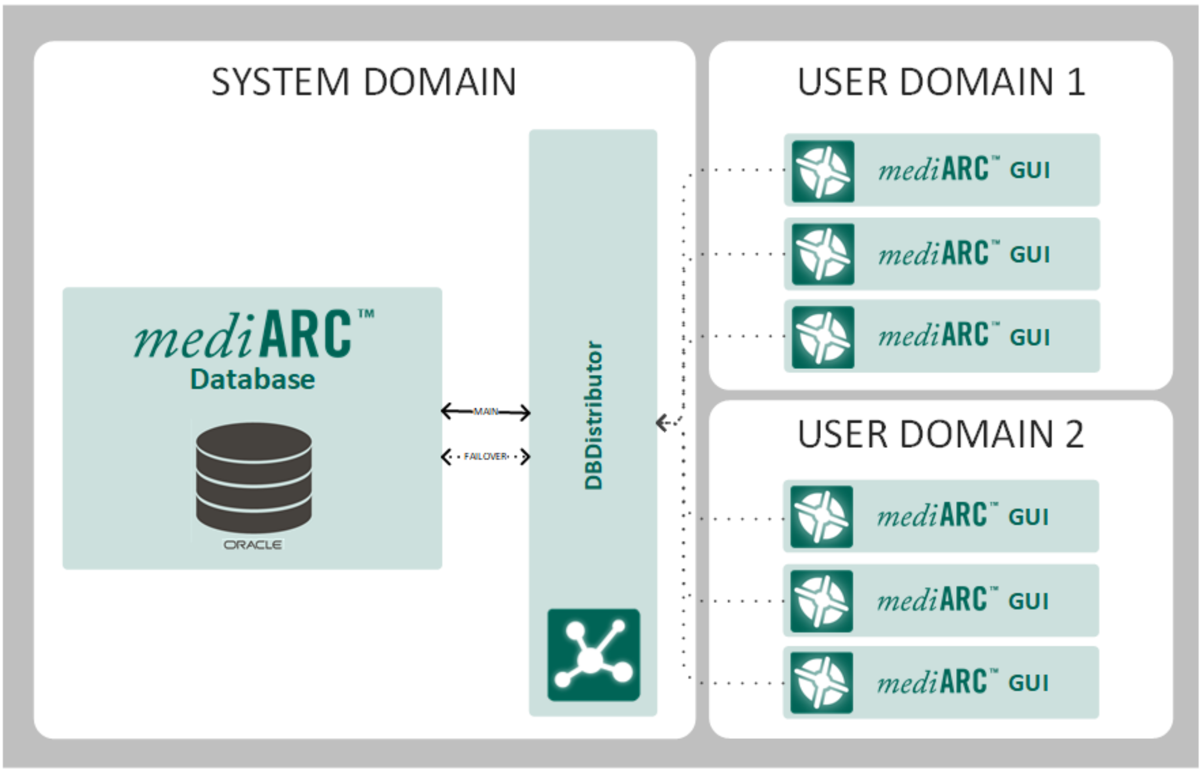
- *ANY.MONITOR*
- *ANY.CONTROL*

# DBDistributor (DBD)

# 5 DBDistributor (DBD)

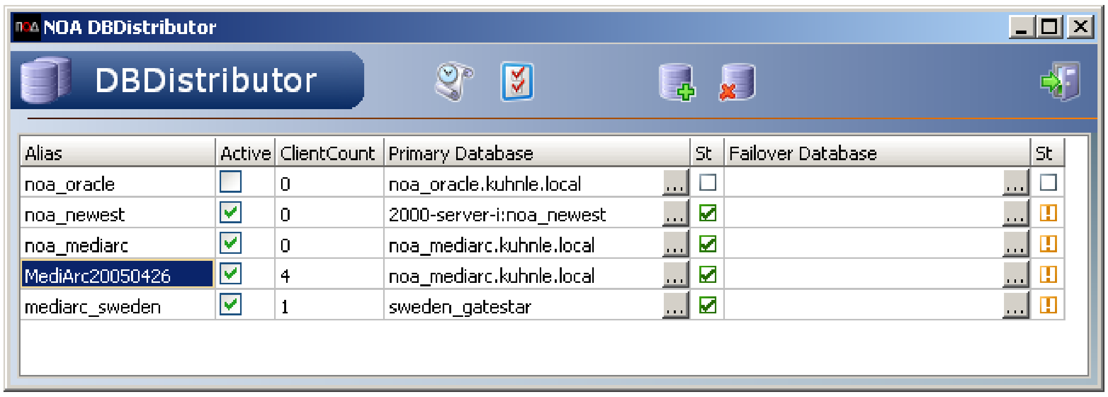
## 5.1 Overview

DBDistributor is used by mediARC GUI as an interface to the mediARC database. It allows to configure database connections with optional fail-over connections.



*A basic DBDistributor setup*

In a mediARC environment the DBDistributor needs to run on a separate Windows server, usually in combination with the LicenseServer and the ProcessorHost.



| Alias | Active | ClientCount | Primary Database | St | Failover Database | St |
|---|---|---|---|---|---|---|
| noa_oracle | ☐ | 0 | noa_oracle.kuhnle.local | ☐ | | ☐ |
| noa_newest | ✓ | 0 | 2000-server-i:noa_newest | ✓ | | ! |
| noa_mediarc | ✓ | 0 | noa_mediarc.kuhnle.local | ✓ | | ! |
| MediArc20050426 | ✓ | 4 | noa_mediarc.kuhnle.local | ✓ | | ! |
| mediarc_sweden | ✓ | 1 | sweden_gatestar | ✓ | | ! |

*DBDistributor Main Window*

In the main window you can create new database connections, view the connection status or delete connections. It is also possible to deactivate a connection temporarily by clicking the checkbox in the active cell.

Following buttons are available:

Show Log: Shows the DBDistributor log window.

Change Ini Data: Shows the application wide settings window.

Add New Database Connection: Adds a new database connection.

Delete Database Connection: Removes a database connection.

*Note that under normal conditions, DBDistributor will be executed as a Windows service. On occasions, it is possible to run DBDistributor as an executable program. Please make sure to stop the service before running the executable.*

*Warning: Stopping the DBDistributor causes all related programs to stop working! Please refer to the mediARC Startup and Shutdown Sequence chapter to plan this step accordingly...*

## 5.2 Installation

To install or reinstall the DBDistributor, simply run the installer and follow the instructions.
During installation you can decide whether you want to install DBDistributor as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

**Prerequisites**

- DBDistributor needs a pre-installed OracleClient (32-bit version) pointing to the right database schema.

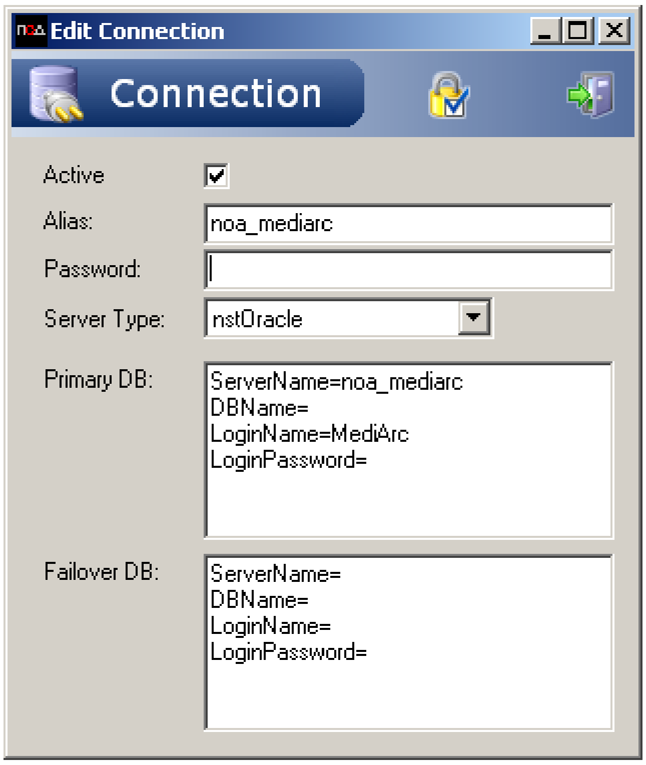
*Note that in 64-bit versions of Windows, the DBDistributor needs to be installed in a directory whose full path does not contain special characters (e.g. C:\NOA\) and not in the default directory "C:\Program Files (x86)\NOA\" because the brackets "(" and ")" will break the connection!*

## 5.3 Configuration

*The configuration data of DBDistributor is stored in the binary 'config.dat' file in the DBDistributor installation directory.*

### 5.3.1 Database Connection Setup

To add a new connection setting, click on the Add New Connection button.
If you want to modify an existing entry you have to click on the ellipsis button ···. The connection detail dialog will appear.



| | |
|---|---|
| Active | ✓ |
| Alias: | noa_mediarc |
| Password: | |
| Server Type: | nstOracle |
| Primary DB: | ServerName=noa_mediarc<br>DBName=<br>LoginName=MediArc<br>LoginPassword= |
| Failover DB: | ServerName=<br>DBName=<br>LoginName=<br>LoginPassword= |

*DBDistributor Connection Configuration*

The dialog window allows you to set the following parameters:

Active: Selects whether the connection is allowed to be used or not. If a connection is inactive, client connections to the database are rejected. Note that existing connections are not dropped if a connection is switched from active to inactive state.
Alias: Database alias name. This name is used by clients to specify which database connection they want to use. The alias must be unique.
Password: If a password is set, only clients with a valid password or clients found in the acceptance list (accessible through the Edit Acceptance List button on the top panel) are accepted.
Server Type: The database server type (Oracle, MS SQL). For mediARC the server type should be set to Oracle.
Primary DB: Connection string for the primary database with one name-value property pair each row.

- Oracle connections use the properties ServerName, LoginName and LoginPassword. Note that the server name represents the TNS alias as set in the Oracle config file `tnsnames.ora`.
- MS SQL connections use the same properties and DBName additionally. Here the server name is the host name of the database server and DBName is the name of the database instance.

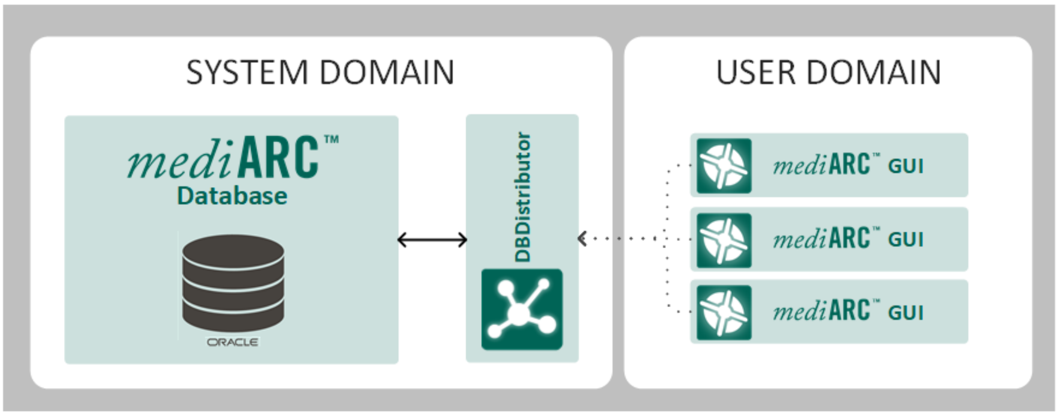
Failover DB: Connection string for failover database connections. The failover connection is used if the primary connection has an error status. For the string syntax see Primary DB above. For more information see the next chapter Failover Database Connection.

*Note that whenever the connection data changes it takes a few seconds until the status icon is updated.*

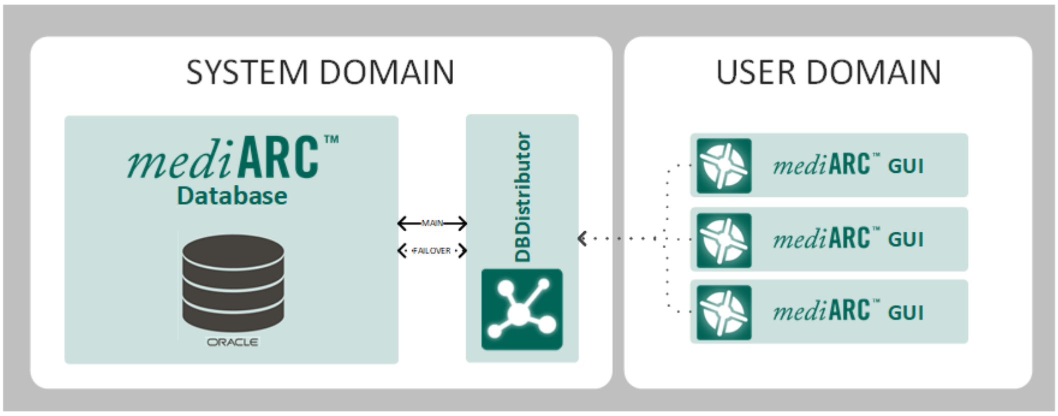
If you want to restrict connections to explicit listed hosts you have to click on the Edit Acceptance List symbol. In the dialog window that will appear you can list the host names either by name or IP number – one host per line. If you don't want to use the acceptance list you can leave it empty. In that case DBDistributor will allow connections from any host on the network.

### 5.3.1.1 Failover Database Connection

The Failover DB connection of DBDistributor is used for setting up an alternative database connection, not an alternative database.
Example configurations (from simplest to most complex):

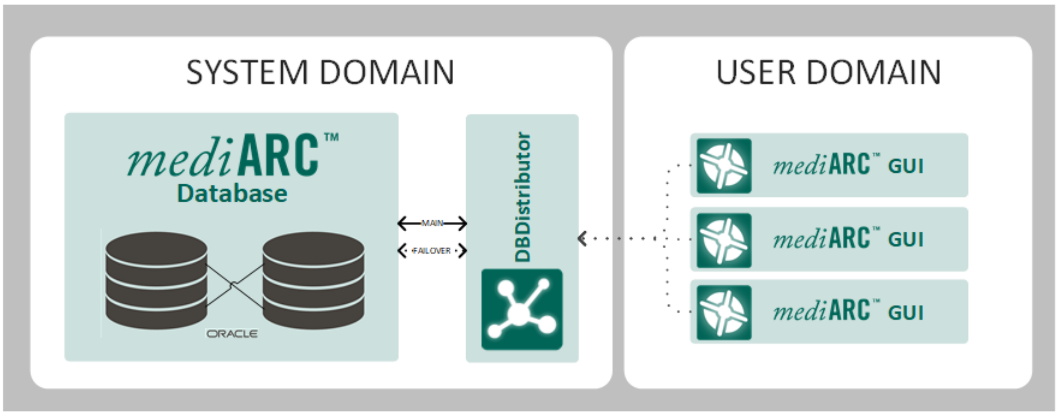


*The most common simple mediARC Database setup*



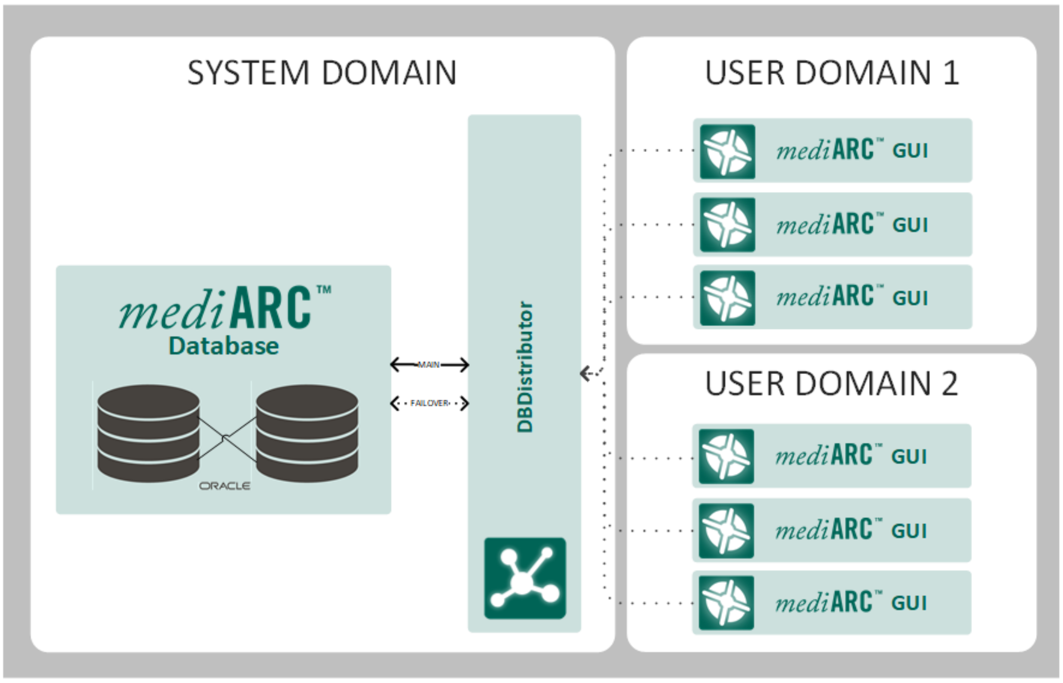
*The same simple mediARC Database setup with a failover connection to the same database*

*Note that mediARC does not provide for any kind of database synchronization!*
*While it is possible to have two separate databases set up with database synchronization, this must be done on Oracle level (RAC) outside of the mediARC environment. The DBDistributor only provides a way to use an alternative connection if the primary DB connection is not available, and does not distinguish between just another DNS name/port to the same DB, or actually a separate database instance with replicated data.*



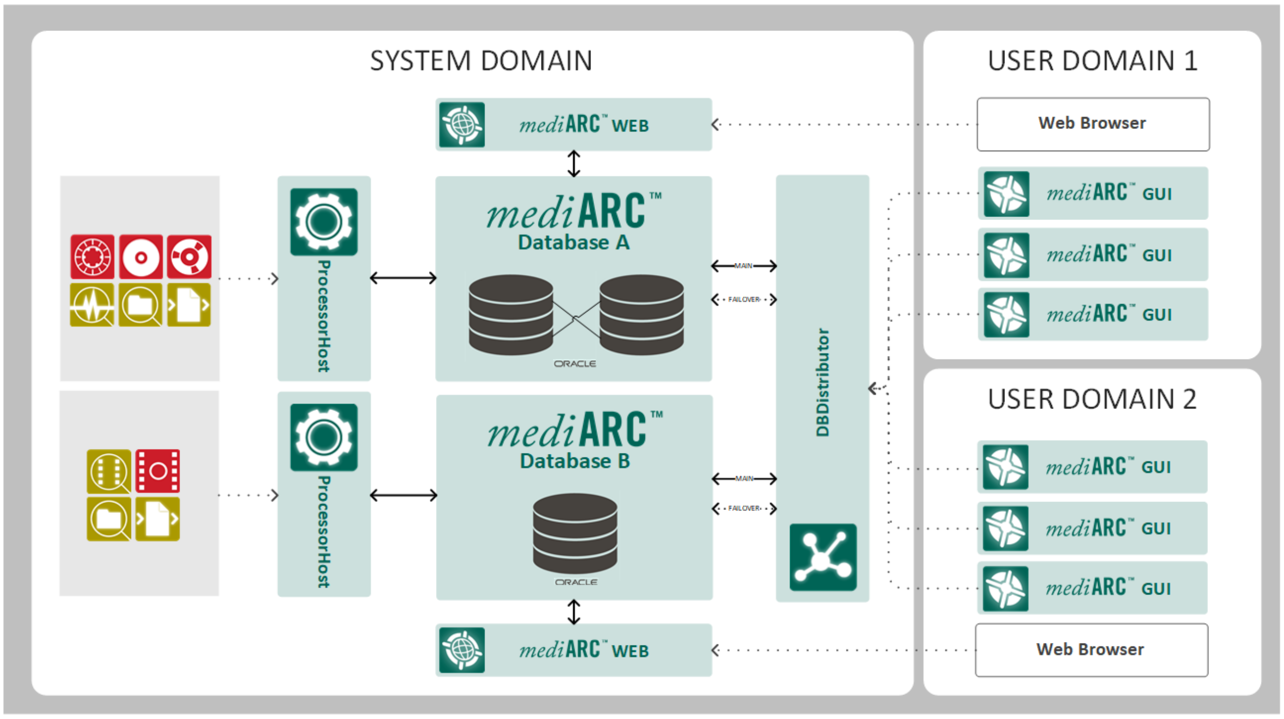
*Because mediARC itself does not provide any kind of database synchronization, from mediARC's perspective this setup is identical to the previous one*

When multiple User Domains are configured within a mediARC system, one DBDistributor is enough to facilitate connections to all users:



*One DBDistributor used for connecting users from various User Domains to the same mediARC Database*

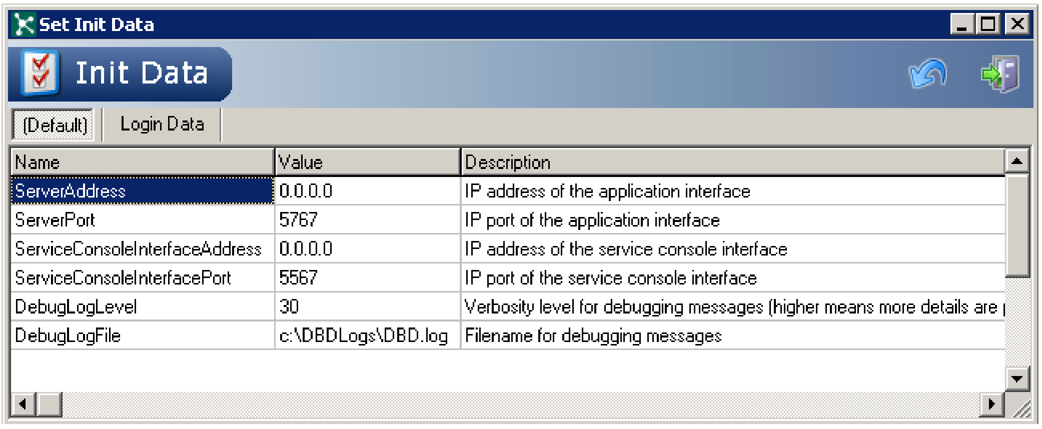
Should more than one mediARC Database be present in a highly complex system, it is possible to define multiple database connections in just one DBDistributor instance. However, other components which do not have this possibility, would have to be multiplied accordingly.



*An example of a highly complex mediARC system setup with multiple databases and multiple User Domains*

## 5.3.2 Init Data Window

There are some application-wide settings that can be found either using the Change Ini Data button or remotely by using the ServiceConsole.



| Name | Value | Description |
|---|---|---|
| ServerAddress | 0.0.0.0 | IP address of the application interface |
| ServerPort | 5767 | IP port of the application interface |
| ServiceConsoleInterfaceAddress | 0.0.0.0 | IP address of the service console interface |
| ServiceConsoleInterfacePort | 5567 | IP port of the service console interface |
| DebugLogLevel | 30 | Verbosity level for debugging messages (higher means more details are |
| DebugLogFile | c:\DBDLogs\DBD.log | Filename for debugging messages |

*The Init Data Window of DBDistributor*

*Note that to restrict database access, it is possible to set a connection password and to set a list of accepted IP addresses.*

The following Init Data settings are used by the DBDistributor:

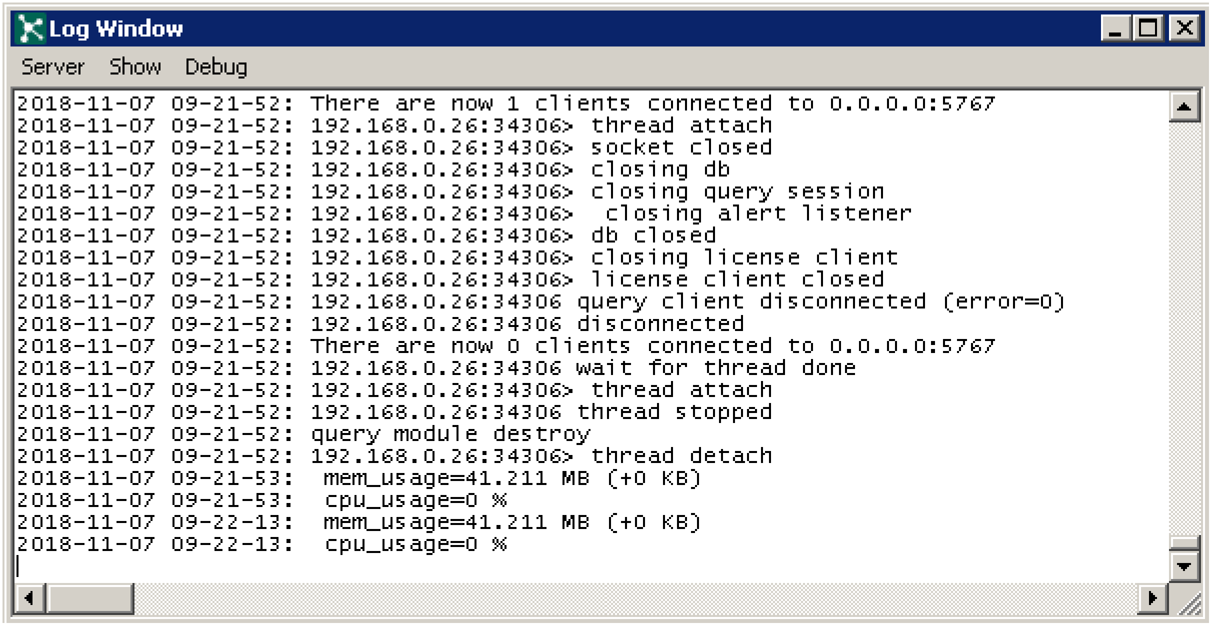
**Default Tab**
ServerAddress: The IP interface address for listening for mediARC GUI connections (may be 0.0.0.0 to listen on any interface). Defaults to "0.0.0.0".
ServerPort: The IP interface port number for listening for mediARC GUI connections. Defaults to "5767".
ServiceConsoleInterfaceAddress: The IP interface address for listening for ServiceConsole connections (may be 0.0.0.0 to listen on any interface). Defaults to "0.0.0.0".
ServiceConsoleInterfacePort: The IP interface port number for listening for ServiceConsole connections. Defaults to "5567".
DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\DBDlogs\DBD.log"). If a file name is provided, debug messages will be written to this file. The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

**Login Data Tab**
LicenseServer: The Hostname or IP address of the LicenseServer. Defaults to "localhost".
LicensePort: IP port of the LicenseServer. Defaults to "5557".

## 5.3.3 Log Window

The Log Window does allow real time observation of DBDistributor activity.



```
2018-11-07 09-21-52: There are now 1 clients connected to 0.0.0.0:5767
2018-11-07 09-21-52: 192.168.0.26:34306> thread attach
2018-11-07 09-21-52: 192.168.0.26:34306> socket closed
2018-11-07 09-21-52: 192.168.0.26:34306> closing db
2018-11-07 09-21-52: 192.168.0.26:34306> closing query session
2018-11-07 09-21-52: 192.168.0.26:34306>  closing alert listener
2018-11-07 09-21-52: 192.168.0.26:34306> db closed
2018-11-07 09-21-52: 192.168.0.26:34306> closing license client
2018-11-07 09-21-52: 192.168.0.26:34306> license client closed
2018-11-07 09-21-52: 192.168.0.26:34306 query client disconnected (error=0)
2018-11-07 09-21-52: 192.168.0.26:34306 disconnected
2018-11-07 09-21-52: There are now 0 clients connected to 0.0.0.0:5767
2018-11-07 09-21-52: 192.168.0.26:34306 wait for thread done
2018-11-07 09-21-52: 192.168.0.26:34306> thread attach
2018-11-07 09-21-52: 192.168.0.26:34306 thread stopped
2018-11-07 09-21-52: query module destroy
2018-11-07 09-21-52: 192.168.0.26:34306> thread detach
2018-11-07 09-21-53:  mem_usage=41.211 MB (+0 KB)
2018-11-07 09-21-53:  cpu_usage=0 %
2018-11-07 09-22-13:  mem_usage=41.211 MB (+0 KB)
2018-11-07 09-22-13:  cpu_usage=0 %
```

*DBDistributor Log Window*

The menu items of the Log Window allow access to following functions:

Server
- Listen (CTRL+L): enables listening to new client connections (new clients will be able to log in to mediARC).
- Close (CTRL+O): disables listening to new client connections (new clients will not be able to log in to mediARC but old clients will not be disconnected).

Show
- Connection List: brings forward the main DBDistributor window.
- Ini data dialog: opens the Init Data window described above.

Debug
- LicenseServer login: tries to log into the LicenseServer as defined in the Login Data tab of the Init Data window and reports the result in the console.
- Enable Encryption: Debugging features for developers. Not relevant for customers.
- Client List: displays inside the console the current status of DBDistributor, the count of currently logged on clients, their IP addresses and Alias of the database they are connected to.
- Test BG Extension: Debugging features for developers. Not relevant for customers.

# mediARC Client Applications

# 6 mediARC Client Applications

mediARC offers two native client applications for users and administrators to access the mediARC system.

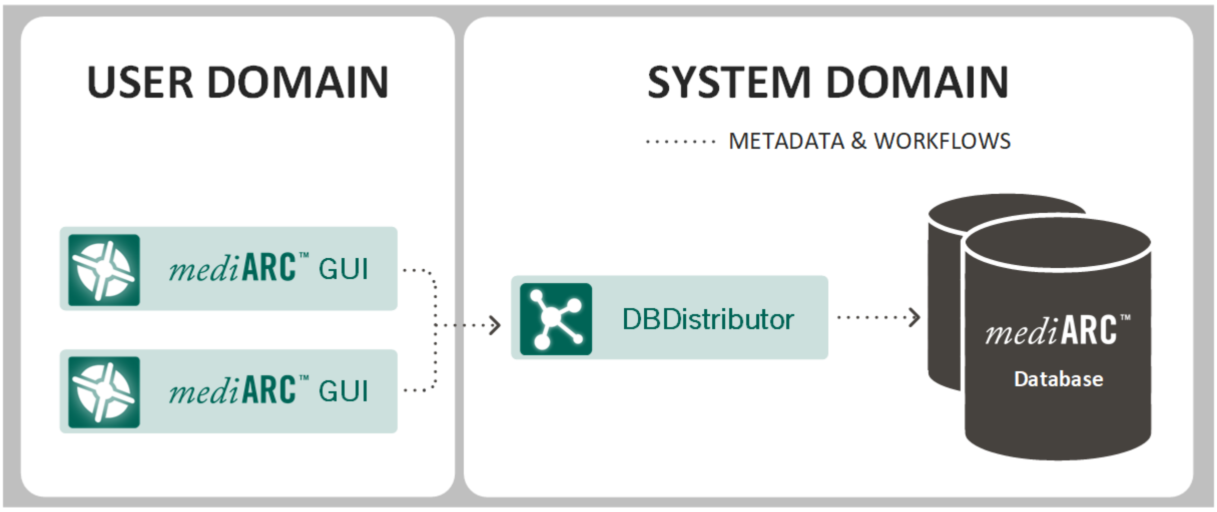
| | |
|---|---|
| mediARC GUI | The full client application that is installed on a local Windows PC. |
| mediARC WEB | A browser based intranet client application with limited features. |

## 6.1 mediARC GUI

The mediARC GUI client is the main interface for users and administrators to access the mediARC system.

Designed with the skilled archivist in mind, the mediARC GUI is a Windows client application with a graphical user interface (GUI) to access and work in the mediARC Archive Asset Management system. The mediARC GUI is the starting point for creating and describing archive objects, modeling and managing workflows, and researching assets within the mediARC database. mediARC is a client-server application used to build an OAIS conform audio and video archive. Media as well as metadata can be managed in mediARC, while workflows handle all archive transactions. The access to the catalogue and archive is carried out via the mediARC GUI within the intranet of an institution.

Each GUI client connects via DBDistributor to the mediARC database.



*mediARC GUI connects to the mediARC Database via DBDistributor*

*For further information, please refer to the mediARC GUI documentation:*

- *for more details regarding operation topics, please consult the mediARC User Manual.*

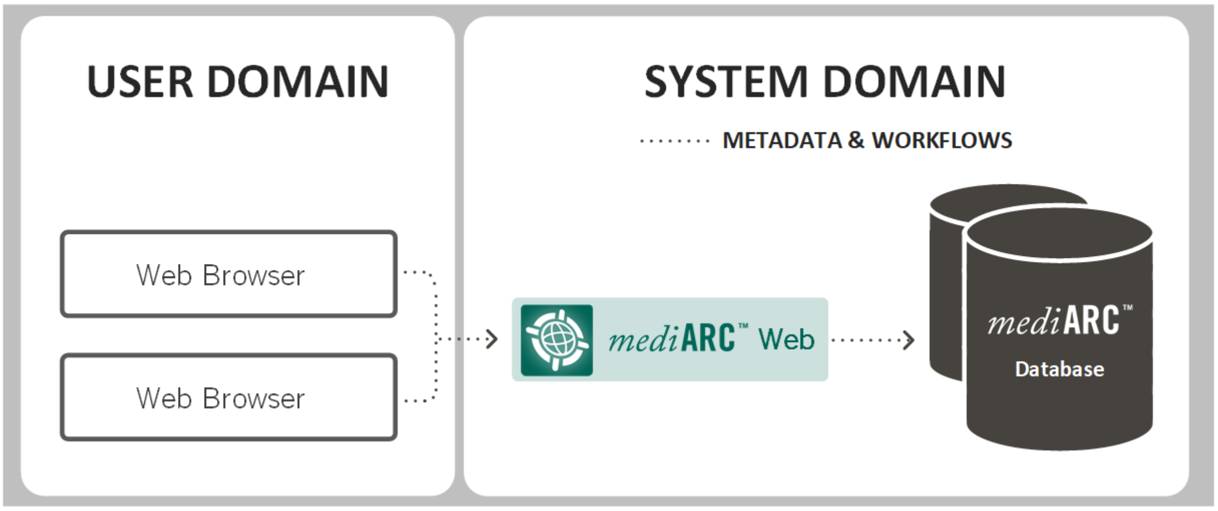
- *for administrative topics, please consult available mediARC administrator documentation.*

## 6.2 mediARC WEB

The mediARC WEB allows users to access the mediARC system via a web browser interface.

mediARC WEB is the intranet web version of the mediARC GUI, which gives a limited set of functionality including search, item details, media prelisten/preview and user orders. It is laid down as a PHP application and may run on different web server operating systems.

mediARC WEB connects directly to the mediARC database.



*User Access via mediARC WEB*

*For further information, please refer to the mediARC WEB documentation.*

# mediARC API (MAPI)

# 7 mediARC API (MAPI)

The mediARC API is a PL/SQL package that provides an API access to the mediARC system. It is designed to be accessed via Oracle database connections or by http via a Web Server.

*Note that a separate license is required to gain access to the mediARC API.*

*For further information, please refer to the mediARC API documentation.*

# ProcessorHost (PH)

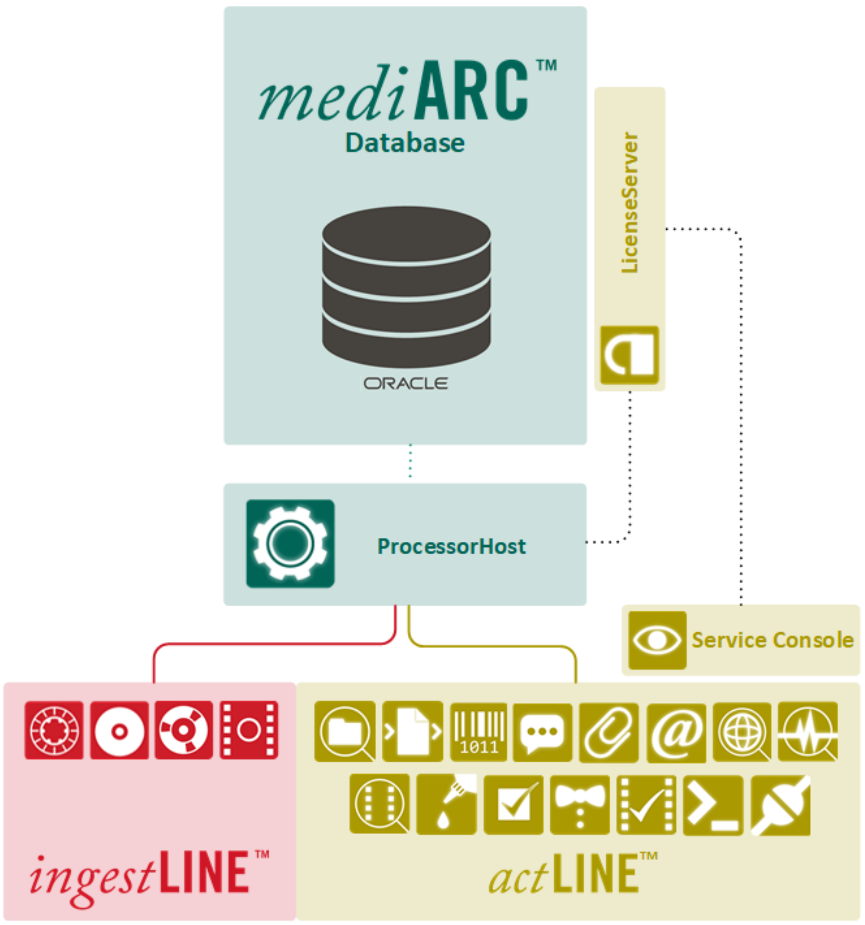
# 8 ProcessorHost (PH)

## 8.1 Overview

ProcessorHost is used as a high-level interface between processors (e.g. ingest applications) and the database. This isolation layer between the database and the processors introduces a client-server scenario, resolving the client requests on a dedicated logical communication layer towards the database.

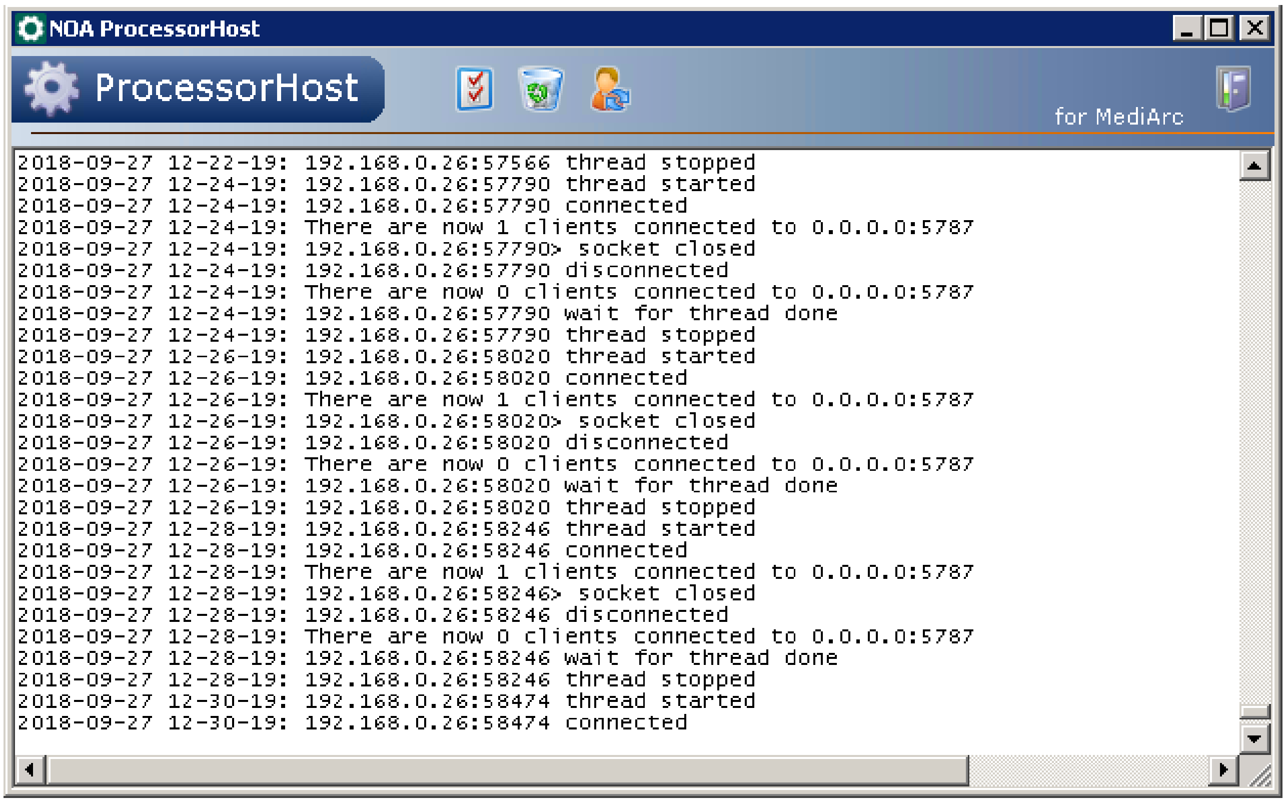
In a mediARC environment the ProcessorHost needs to run on a separate Windows server, usually in combination with the LicenseServer and the DBDistributor.

*As a server, ProcessorHost allows the processor applications to access the mediARC database using high-level* NoaComm *remote procedure calls, reducing the communication need and providing a layer of compatibility between different database versions.*



*ingestLine And actLINE Processors Connect Over ProcessorHost To mediARC Database*

*ProcessorHost is existing for jobDB or mediARC environment, thus encapsulating the processor applications from their current environment system. This allows customers to smoothly upgrade from jobDB to mediARC with no big difference on the client side.*



```
2018-09-27 12-22-19: 192.168.0.26:57566 thread stopped
2018-09-27 12-24-19: 192.168.0.26:57790 thread started
2018-09-27 12-24-19: 192.168.0.26:57790 connected
2018-09-27 12-24-19: There are now 1 clients connected to 0.0.0.0:5787
2018-09-27 12-24-19: 192.168.0.26:57790> socket closed
2018-09-27 12-24-19: 192.168.0.26:57790 disconnected
2018-09-27 12-24-19: There are now 0 clients connected to 0.0.0.0:5787
2018-09-27 12-24-19: 192.168.0.26:57790 wait for thread done
2018-09-27 12-24-19: 192.168.0.26:57790 thread stopped
2018-09-27 12-26-19: 192.168.0.26:58020 thread started
2018-09-27 12-26-19: 192.168.0.26:58020 connected
2018-09-27 12-26-19: There are now 1 clients connected to 0.0.0.0:5787
2018-09-27 12-26-19: 192.168.0.26:58020> socket closed
2018-09-27 12-26-19: 192.168.0.26:58020 disconnected
2018-09-27 12-26-19: There are now 0 clients connected to 0.0.0.0:5787
2018-09-27 12-26-19: 192.168.0.26:58020 wait for thread done
2018-09-27 12-26-19: 192.168.0.26:58020 thread stopped
2018-09-27 12-28-19: 192.168.0.26:58246 thread started
2018-09-27 12-28-19: 192.168.0.26:58246 connected
2018-09-27 12-28-19: There are now 1 clients connected to 0.0.0.0:5787
2018-09-27 12-28-19: 192.168.0.26:58246> socket closed
2018-09-27 12-28-19: 192.168.0.26:58246 disconnected
2018-09-27 12-28-19: There are now 0 clients connected to 0.0.0.0:5787
2018-09-27 12-28-19: 192.168.0.26:58246 wait for thread done
2018-09-27 12-28-19: 192.168.0.26:58246 thread stopped
2018-09-27 12-30-19: 192.168.0.26:58474 thread started
2018-09-27 12-30-19: 192.168.0.26:58474 connected
```

*mediARC ProcessorHost*

Following buttons are available:

Change Ini-Data: Shows the application wide settings window.

Show Garbage Collector: Shows the Garbage Collector window.

Show User Sync: Shows the UserSync window.

*Note that under normal conditions, ProcessorHost will be executed as a Windows service. On occasions, it is possible to run ProcessorHost as an executable program. Please make sure to stop the service before running the executable.*

Besides facilitating a connection to the mediARC database, the ProcessorHost also takes care of the following:

- centralizing all client logs by writing the processor log files for all clients into one specific location
- managing the system's central storage (see General Public Path)
- purging files on the General Public Path after a certain time out (see GarbageCollector).

*Warning: Stopping the ProcessorHost causes all related programs and processes to stop working! Please refer to the mediARC Startup and Shutdown Sequence chapter to plan this step accordingly...*

## 8.2 Installation

To install or reinstall the ProcessorHost, simply run the installer and follow the instructions.
During installation you can decide whether you want to install ProcessorHost as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

**Prerequisites**

- ProcessorHost needs a pre-installed OracleClient (32-bit version) pointing to the right database schema.
- Because ProcessorHost connects to LicenseServer it is necessary to install and start LicenseServer before running ProcessorHost.
- It is recommended to install some HTTP web server (e.g. Microsoft IIS) to allow users browsing the HTML log files.

*Note that in 64-bit versions of Windows, the ProcessorHost needs to be installed in a directory whose full path does not contain special characters (e.g. C:\NOA\) and not in the default directory "C:\Program Files (x86)\NOA\" because the brackets "(" and ")" will break the connection!*

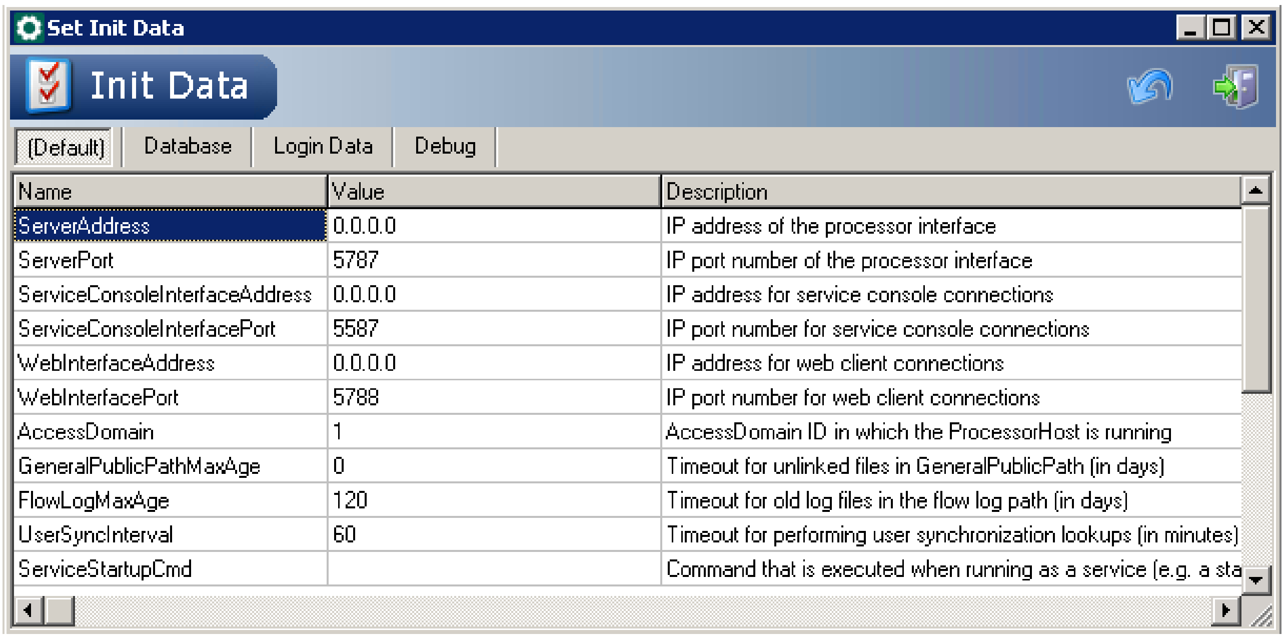
## 8.3 Configuration

*The configuration data of ProcessorHost is stored in the Windows registry, thus it is available for the service as well as for the application. Please take care which Windows user account you use when configuring the application as the settings will be available only for this user account. This is important when running the service version later on because the service must then be configured to use the same user account which was used during the application configuration step.*

*The registry entries can be found in HKEY_CURRENT_USER\Software\NOA\MedPrcHost.*

### 8.3.1 Init Data Window

There are some application-wide settings that can be found either using the Change Ini Data button or remotely by using the ServiceConsole.



| Name | Value | Description |
|---|---|---|
| ServerAddress | 0.0.0.0 | IP address of the processor interface |
| ServerPort | 5787 | IP port number of the processor interface |
| ServiceConsoleInterfaceAddress | 0.0.0.0 | IP address for service console connections |
| ServiceConsoleInterfacePort | 5587 | IP port number for service console connections |
| WebInterfaceAddress | 0.0.0.0 | IP address for web client connections |
| WebInterfacePort | 5788 | IP port number for web client connections |
| AccessDomain | 1 | AccessDomain ID in which the ProcessorHost is running |
| GeneralPublicPathMaxAge | 0 | Timeout for unlinked files in GeneralPublicPath (in days) |
| FlowLogMaxAge | 120 | Timeout for old log files in the flow log path (in days) |
| UserSyncInterval | 60 | Timeout for performing user synchronization lookups (in minutes) |
| ServiceStartupCmd |  | Command that is executed when running as a service (e.g. a sta |

*mediARC ProcessorHost Init Data Window*

The following Init Data settings are used by the ProcessorHost:

**Defaul Tab**

ServerAddress: The IP interface address for listening for processor application connections. The special value "0.0.0.0" may be used to listen on all available interfaces. Defaults to "0.0.0.0".
ServerPort: The IP interface port number for listening for processor application connections. Defaults to "5787".
ServiceConsoleInterfaceAddress: The IP interface address for listening for ServiceConsole connections. The special value "0.0.0.0" may be used to listen on all available interfaces. Defaults to "0.0.0.0".
ServiceConsoleInterfacePort: The IP interface port number for listening for ServiceConsole connections. Defaults to "5587".
WebInterfaceAddress: The IP address for web connections (Flow Logs etc.).
WebInterfacePort: The IP port number for web connections.
AccessDomain: The Access Domain ID in which ProcessorHost is running. For running systems the number can be found in the Oracle database (`SELECT * FROM AccessDomain`).
GeneralPublicPathMaxAge: The timeout for unlinked files on the General Public Path (in days). If the timeout value is zero, the clean-up of unlinked files is disabled. Defaults to "0" (disabled). See GarbageCollector.
FlowLogMaxAge: The timeout for old log files in the flow log path (in days). If the timeout value is zero, the clean-up of old log files is disabled. Defaults to "120".
UserSyncInterval: The timeout for performing user synchronization lookups (in minutes). Defaults to "0" (disabled). See UserSync.
ServiceStartupCmd: The command that is executed when running as a service (e.g. a startup script). Leave blank if not used.

**Database Tab**

DBServer: The TNS Alias of the Oracle database connection.
DBUsername: Login credentials for the database server.
DBPassword: Login credentials for the database server.

**Login Data Tab**
LicenseServer: The hostname or IP address of the LicenseServer connection. Defaults to "localhost".
LicensePort: The IP port of the LicenseServer connection. Defaults to "5557".

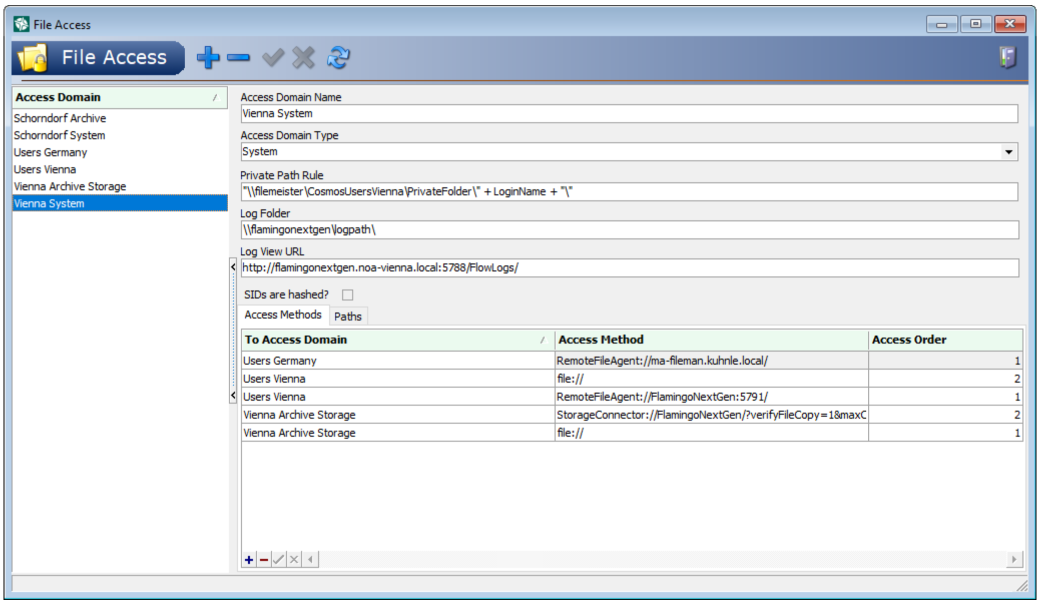
**Debug Tab**
DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\PHlogs\PH.log"). If a file name is provided, debug messages will be written to this file. A new log file is started automatically by ProcessorHost if the existing log file exceeds a size limit (currently 20MB). The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".
TaskLogLevel: The level of detail for task log messages (on a scale from 1 to 5, where 5 represents the most detailed logging). Task logs document task execution and are handled centralized through the ProcessorHost. Depending on the level, bare execution or even detailed parameters used are logged. In contrast to debug logging, task logging is part of normal operation. It is recommended to use a more detailed logging when changes are introduced to the process. As soon as the changed process is running smooth, task logging may revert to a medium or low detailed level. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

*Note that not only flows and tasks produce log files. There are also log files from the GarbageCollector in the ProcessorHost subfolder and from DBScripter's cyclic scripts in the DBScripter subfolder. ProcessorHost will create subfolders within the log folder for the different processor applications as necessary. The name of the log files is prefixed with* *the date and time to allow sorting out and deleting old log files easily. Log files are moved to a compressed zip-file by the GarbageCollector after a timeout (usually 120 days).*

### 8.3.1.1 Task Log Configuration

The folder where the Task log files are stored by ProcessorHost can be configured in mediARC GUI using the menu Administration | File Access.



| Access Domain |
|---|
| Schorndorf Archive |
| Schorndorf System |
| Users Germany |
| Users Vienna |
| Vienna Archive Storage |
| Vienna System |

| | |
|---|---|
| Access Domain Name | Vienna System |
| Access Domain Type | System |
| Private Path Rule | "\\filemeister\CosmosUsersVienna\PrivateFolder\" + LoginName + "\" |
| Log Folder | \\flamingonextgen\logpath\ |
| Log View URL | http://flamingonextgen.noa-vienna.local:5788/FlowLogs/ |
| SIDs are hashed? | ☐ |

| To Access Domain | Access Method | Access Order |
|---|---|---|
| Users Germany | RemoteFileAgent://ma-fileman.kuhnle.local/ | 1 |
| Users Vienna | file:// | 2 |
| Users Vienna | RemoteFileAgent://FlamingoNextGen:5791/ | 1 |
| Vienna Archive Storage | StorageConnector://FlamingoNextGen/?verifyFileCopy=1&maxC | 2 |
| Vienna Archive Storage | file:// | 1 |

*The File Access Dialog In mediARC GUI*

The two parameters Log Folder and Log View URL are part of the Access Domain, for which the ProcessorHost is configured (see Init Data parameter AccessDomain above).
For backwards compatibility both parameters are read from the mediARC database if no access domain specific values are set. The entries used in this case are:

LogFolder: The path for storing the Task log files. Defaults to "C:\NOA\FlowLogs\".
LogViewUrl: The URL for mediARC GUI clients to view the task log files. Defaults to "file:///C:/NOA/FlowLogs/".

*Note that when the value for Log Folder or Log View URL in the mediARC GUI is changed, the ProcessorHost must be restarted for the changes to take effect!*

*To allow users to view the processor log files more easily, it is recommended to setup an HTTP web server (e.g. Microsoft IIS) which is running on the same machine as ProcessorHost or on any machine that has direct access to the log folder. The log view*

*URL is then configured in a way that a web browser can access the log files via the web server using the specified URL (e.g. http://server/FlowLogs/).*

### 8.3.2 Verify Connection

To see if the connection was successful and if the ProcessorHost is up and running, double click the ProcessorHost icon in the windows tray. The main ProcessorHost window will appear, where, after a fresh ProcessorHost start, you should be able to see entries like this :

```
NOA ProcessorHost Version x.x.x (JobDB, Build #xx)
Starting processor application interface...
Starting service console interface...
Server on <servername>:5587 started.
Connected to database "<DBServername:DatabaseName>"
DB Version is: xx
Outputting Task- and FlowLog to: O:\Archive\Logs
URL to view Task- and FlowLog is: file:///O:/Archive/Logs
Server on <ServerName:Port> started.
```

If the last line states "**`Server on 'ServerName:Port' started`**", ProcessorHost started up without problems and is now ready for use.
If not, read through the entries and fine out about the errors encountered during startup.

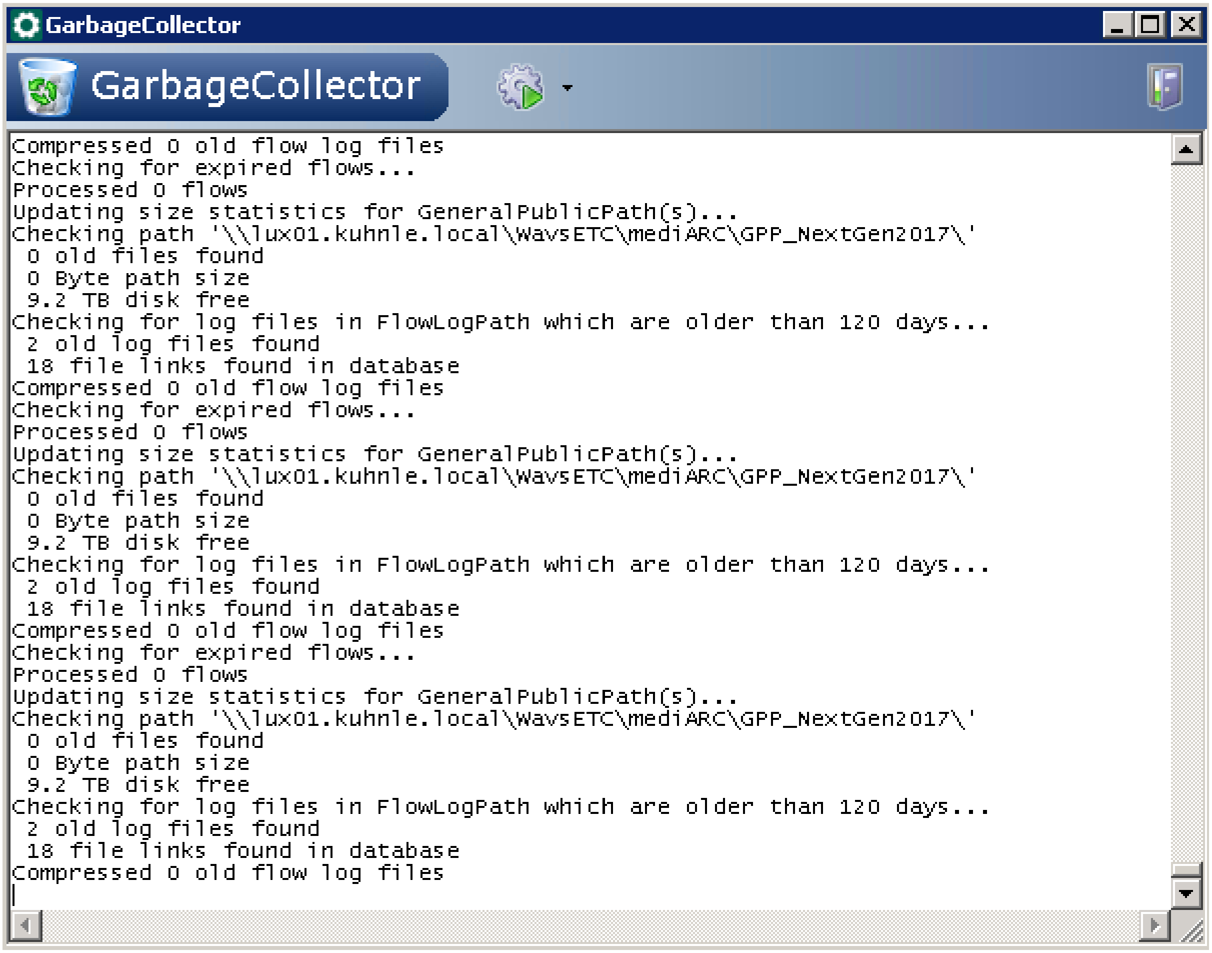
To see if an already running ProcessorHost is responsive, open the main ProcessorHost window in the same way and see if there are new entries being logged. Even with no clients connected, the ProcessorHost logs idle routines in cycles as well.

## 8.4 GarbageCollector

Files which were created by processor applications on the General Public Path can be deleted automatically (in cycles) by a process called Garbage Collection.

If the GeneralPublicPathMaxAge parameter in the Default tab of the Init Data window of ProcessorHost is set to a number larger than 0, files which were created by processor applications and are not linked to a workflow are deleted when their age exceeds the value configured (files which were created by processor applications and linked to a workflow will be deleted immediately when the time out for the parent workflow is reached).

The Garbage Collector window can be accessed using the Show Garbage Collector 🗑 button on the top panel of the ProcessorHost main window.



```
Compressed 0 old flow log files
Checking for expired flows...
Processed 0 flows
Updating size statistics for GeneralPublicPath(s)...
Checking path '\\lux01.kuhnle.local\WavsETC\mediARC\GPP_NextGen2017\'
 0 old files found
 0 Byte path size
 9.2 TB disk free
Checking for log files in FlowLogPath which are older than 120 days...
 2 old log files found
 18 file links found in database
Compressed 0 old flow log files
Checking for expired flows...
Processed 0 flows
Updating size statistics for GeneralPublicPath(s)...
Checking path '\\lux01.kuhnle.local\WavsETC\mediARC\GPP_NextGen2017\'
 0 old files found
 0 Byte path size
 9.2 TB disk free
Checking for log files in FlowLogPath which are older than 120 days...
 2 old log files found
 18 file links found in database
Compressed 0 old flow log files
Checking for expired flows...
Processed 0 flows
Updating size statistics for GeneralPublicPath(s)...
Checking path '\\lux01.kuhnle.local\WavsETC\mediARC\GPP_NextGen2017\'
 0 old files found
 0 Byte path size
 9.2 TB disk free
Checking for log files in FlowLogPath which are older than 120 days...
 2 old log files found
 18 file links found in database
Compressed 0 old flow log files
```

*ProcessorHost Garbage Collector Window*

The following buttons are available:

Run Check Now: Runs the check immediately instead of waiting for the next scheduled run cycle.

The drop down menu of the Run Check Now button additionally offers following options:
Check Now: Runs the check immediately instead of waiting for the next scheduled run cycle. The default action (bold) that is triggered when the button is clicked.
Check Flow Logs: Runs a subset of the task and checks only old flow log files.
Disable Check Timer: Toggles on/off the timer that runs the checks in cycles on automatic schedule.

*Note that if the GeneralPublicPathMaxAge is set to zero, the Garbage Collection of unlinked files is disabled completely.*

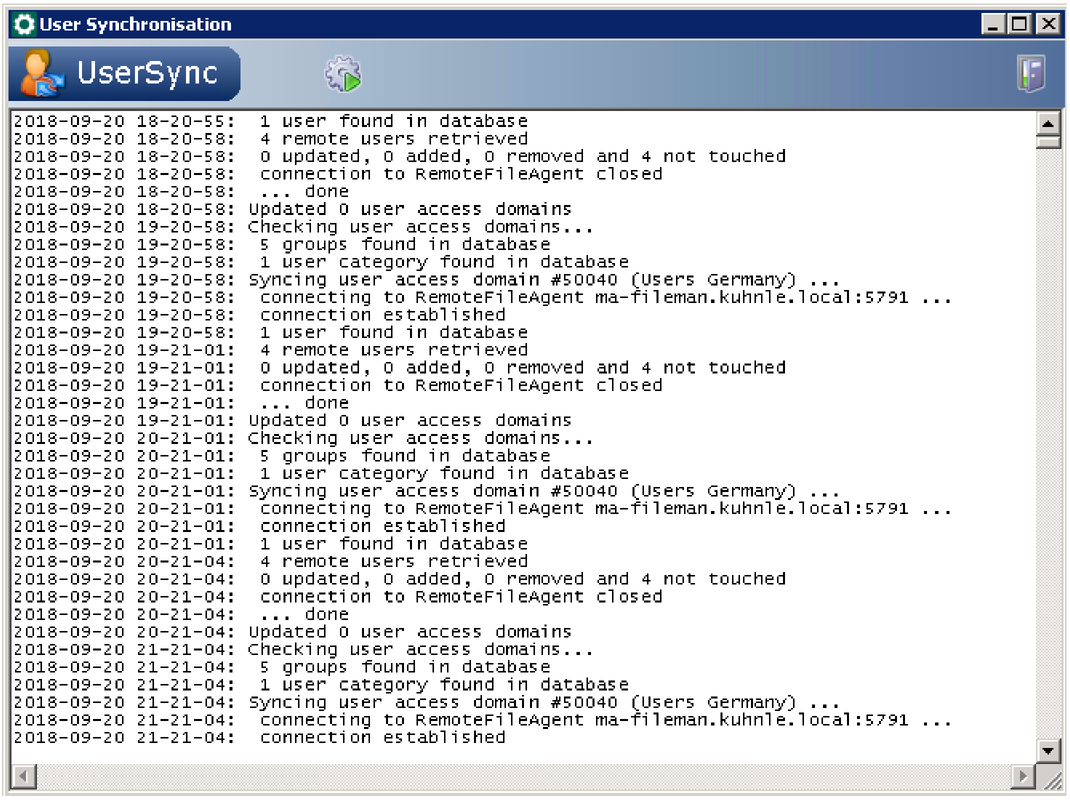
## 8.5 User Synchronization

The user synchronization process allows to synchronize users from external directory services (Active Directory, LDAP) with the mediARC user base.

To allow automatic synchronization, the UserSyncInterval parameter of ProcessorHost needs to be set to a meaningful value (e.g. 60 - 240 minutes). Also the target user Access Domains that should be synced in mediARC need to be configured with access domain type "User" and there needs to be at least one RemoteFileAgent:// access method defined from the System Domain (where ProcessorHost is running) towards the User Domain.

If synchronization is enabled, the ProcessorHost will periodically connect to all the RemoteFileAgents that are configured as Access Methods in the System Domain (that means a RemoteFileAgent:// method is defined from System Domain to User Domain) and send a list of all groups defined in mediARC to them periodically. RemoteFileAgents will then look-up users in matching groups and return a list of user details. ProcessorHost will then compare the list of returned users with the users found in the local database.

In ProcessorHost, the User Synchronization window can be accessed using the Show User Sync... button.



```
2018-09-20 18-20-55:  1 user found in database
2018-09-20 18-20-58:  4 remote users retrieved
2018-09-20 18-20-58:  0 updated, 0 added, 0 removed and 4 not touched
2018-09-20 18-20-58:  connection to RemoteFileAgent closed
2018-09-20 18-20-58:  ... done
2018-09-20 18-20-58: Updated 0 user access domains
2018-09-20 19-20-58: Checking user access domains...
2018-09-20 19-20-58:  5 groups found in database
2018-09-20 19-20-58:  1 user category found in database
2018-09-20 19-20-58: Syncing user access domain #50040 (Users Germany) ...
2018-09-20 19-20-58:  connecting to RemoteFileAgent ma-fileman.kuhnle.local:5791 ...
2018-09-20 19-20-58:  connection established
2018-09-20 19-20-58:  1 user found in database
2018-09-20 19-21-01:  4 remote users retrieved
2018-09-20 19-21-01:  0 updated, 0 added, 0 removed and 4 not touched
2018-09-20 19-21-01:  connection to RemoteFileAgent closed
2018-09-20 19-21-01:  ... done
2018-09-20 19-21-01: Updated 0 user access domains
2018-09-20 20-21-01: Checking user access domains...
2018-09-20 20-21-01:  5 groups found in database
2018-09-20 20-21-01:  1 user category found in database
2018-09-20 20-21-01: Syncing user access domain #50040 (Users Germany) ...
2018-09-20 20-21-01:  connecting to RemoteFileAgent ma-fileman.kuhnle.local:5791 ...
2018-09-20 20-21-01:  connection established
2018-09-20 20-21-01:  1 user found in database
2018-09-20 20-21-04:  4 remote users retrieved
2018-09-20 20-21-04:  0 updated, 0 added, 0 removed and 4 not touched
2018-09-20 20-21-04:  connection to RemoteFileAgent closed
2018-09-20 20-21-04:  ... done
2018-09-20 20-21-04: Updated 0 user access domains
2018-09-20 21-21-04: Checking user access domains...
2018-09-20 21-21-04:  5 groups found in database
2018-09-20 21-21-04:  1 user category found in database
2018-09-20 21-21-04: Syncing user access domain #50040 (Users Germany) ...
2018-09-20 21-21-04:  connecting to RemoteFileAgent ma-fileman.kuhnle.local:5791 ...
2018-09-20 21-21-04:  connection established
```

*ProcessorHost User Sync Window*

The UserSync window allows real time observation of ProcessorHost user synchronization activity.

```
2021-03-09 14-53-10: Checking user access domains...
2021-03-09 14-53-10:  6 groups found in database
2021-03-09 14-53-10:  2 user categories found in database
2021-03-09 14-53-10: Syncing user access domain #50040 (<domain name>) ...
2021-03-09 14-53-10:  connecting to RemoteFileAgent <hostname>:5791 ...
2021-03-09 14-53-10:  connection established
2021-03-09 14-53-10:  14 users found in database
```

It is possible to trigger the synchronization procedure manually by clicking the Run Sync Now .

If UserSyncInterval is set to zero, no automatic user synchronization is performed.

*For more information on user synchronization, see the mediARC User Configuration and Rights Management Administrator Manual.*

# FileManager (FM)

# 9 FileManager (FM)

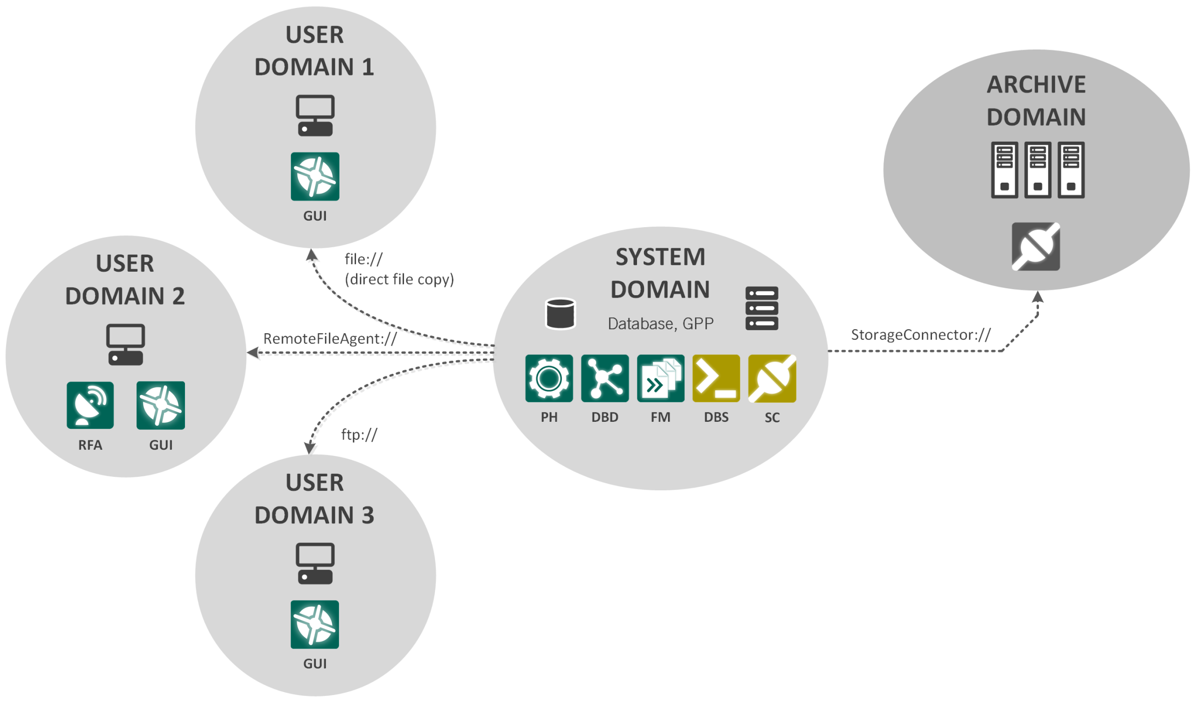
## 9.1 Overview

The FileManager moves files between different Access Domains, selecting the appropriate access method, by using dedicated policies such as:

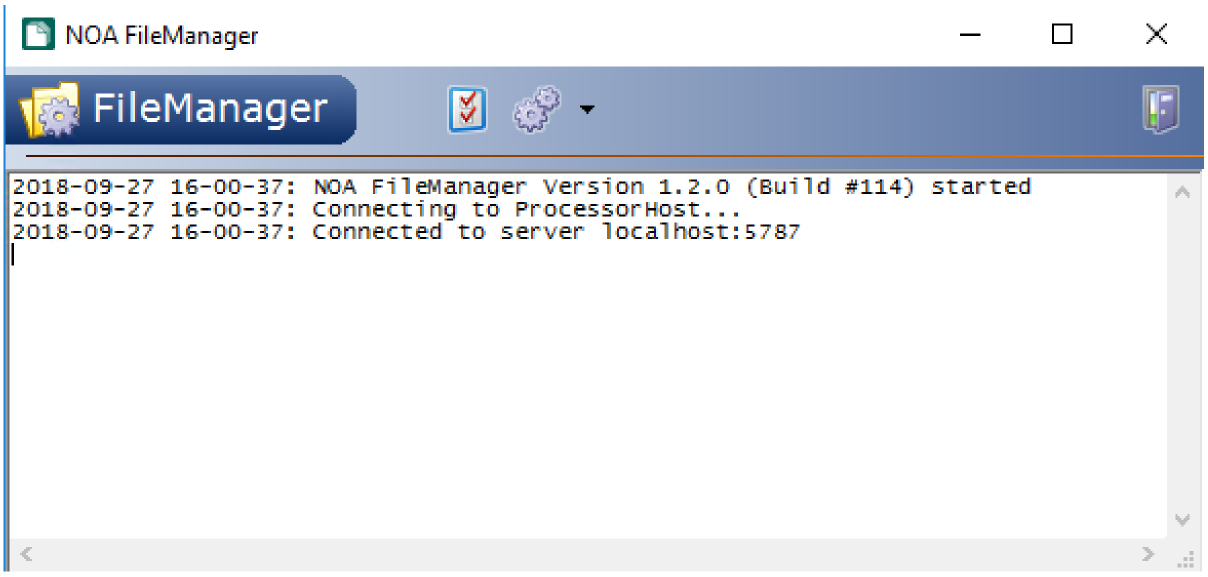
- File system based copy
- Connecting to RemoteFileAgent
- Connecting to StorageConnector
- Connecting to a FTP server

Whenever a file is asked from different Access Domains, the FileManager is tasked with providing the requested files from/to the defined Access Domains. The FileManager represents the link between mediARC's

- User Domains
- System Domain
- Archive Domain



These Access Domains are addressed by FileManager either directly or over RemoteFileAgent and StorageConnector (interface application for FileManager to address specific storage solutions).



```
2018-09-27 16-00-37: NOA FileManager Version 1.2.0 (Build #114) started
2018-09-27 16-00-37: Connecting to ProcessorHost...
2018-09-27 16-00-37: Connected to server localhost:5787
```

*File Delivery by FileManager*

The following buttons are available:

Change Init-Data: Shows the application wide settings window.

Show Processors: Shows the instances of FileManager in separate windows (see the FileManager Instances chapter).

The drop down menu of the Show Processors button additionally offers following options:

- Auto Load: FileManager will load tasks automatically if enabled.
- OnlyReserved: FileManager will only load reserved tasks if enabled.

*Note that under normal conditions, FileManager will be executed as a Windows service. On occasions, it is possible to run FileManager as an executable program. Please make sure to stop the service before running the executable.*

## 9.2 Installation

To install or reinstall the FileManager, simply run the installer and follow the instructions.
During installation you can decide whether you want to install FileManager as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

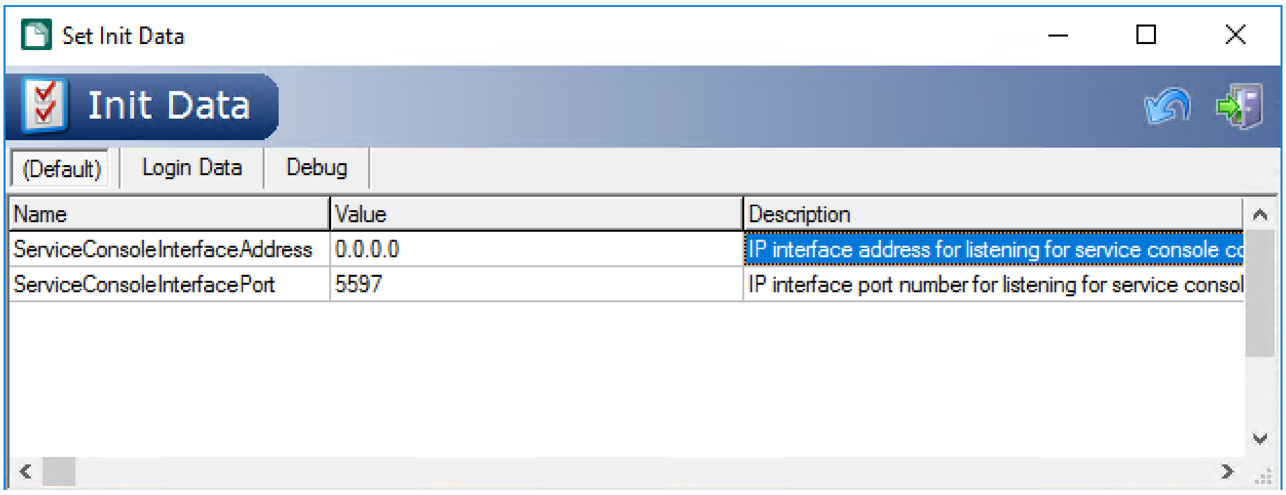
## 9.3 Configuration

*The configuration data of FileManager is stored in the Windows registry, thus it is available for the service as well as for the application. Please take care which Windows user account you use when configuring the application as the settings will be available only for this user account. This is important when running the service version later on because the service must then be configured to use the same user account which was used during the application configuration step.*

*The registry entries can be found in HKEY_CURRENT_USER\SOFTWARE\NOA\FileManager.*

### 9.3.1 Init Data Window

There are some application-wide settings that can be found either using the Change Ini-Data button or remotely by using the ServiceConsole.



| Name | Value | Description |
|---|---|---|
| ServiceConsoleInterfaceAddress | 0.0.0.0 | IP interface address for listening for service console co |
| ServiceConsoleInterfacePort | 5597 | IP interface port number for listening for service consol |

*mediARC FileManager Init Data Window*

The following Init Data settings are used by the FileManager:

**Default Tab**
ServiceConsoleInterfaceAddress: The IP interface address for listening for service console connections (may be 0.0.0.0 to listen on any interface). Defaults to "0.0.0.0".
ServiceConsoleInterfacePort: The IP interface port number for listening for service console connections. Defaults to "5597".

**Login Data Tab**
ProcessorHost: The IP address of the ProcessorHost.
ProcessorPort: The IP port of the ProcessorHost.

**Debug Tab**

DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\FMlogs\FM.log"). If a file name is provided, debug messages will be written to this file. The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".
TaskLogLevel: The level of detail for task log messages (on a scale from 1 to 5, where 5 represents the most detailed logging). Task logs document task execution and are handled centralized through the ProcessorHost. Depending on the level, bare execution or even detailed parameters used are logged. In contrast to debug logging, task logging is part of normal operation. It is recommended to use a more detailed logging when changes are introduced to the process. As soon as the changed process is running smooth, task logging may revert to a medium or low detailed level. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

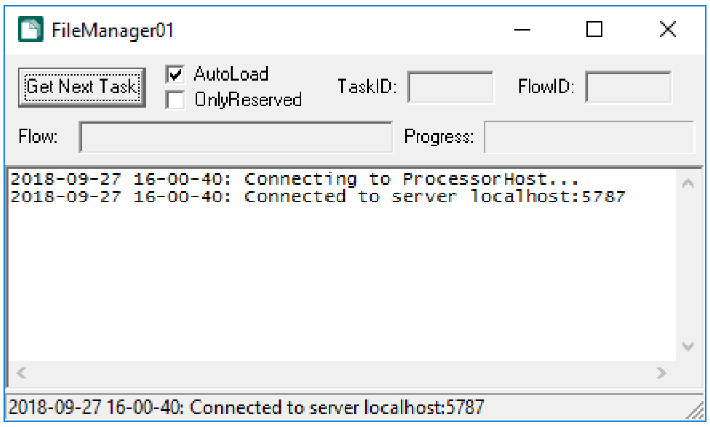
### 9.3.2 mediARC File Access

Note that Access Methods and Paths that are required by the FileManager need to be defined in mediARC GUI using the menu Administration | File Access.

*For further information about File Access configuration in mediARC, please refer to the mediARC Domains TechNote or the mediARC User Configuration and Rights Management Administrator Manual.*

## 9.4 FileManager Instances

FileManager can be multi-instanced. When running FileManager as a GUI application, each instance can be opened as a single processor window by clicking the Show Processors button in the FileManager main window.



```
2018-09-27 16-00-40: Connecting to ProcessorHost...
2018-09-27 16-00-40: Connected to server localhost:5787
```

2018-09-27 16-00-40: Connected to server localhost:5787

*FileManager Single Processor Window*

FileManager Tasks can be loaded manually or automatically, and a Processor can be configured to only load reserved Tasks.

*For further information about Processors, please refer to the mediARC Computers and Processors TechNote.*

# RemoteFileAgent (RFA)

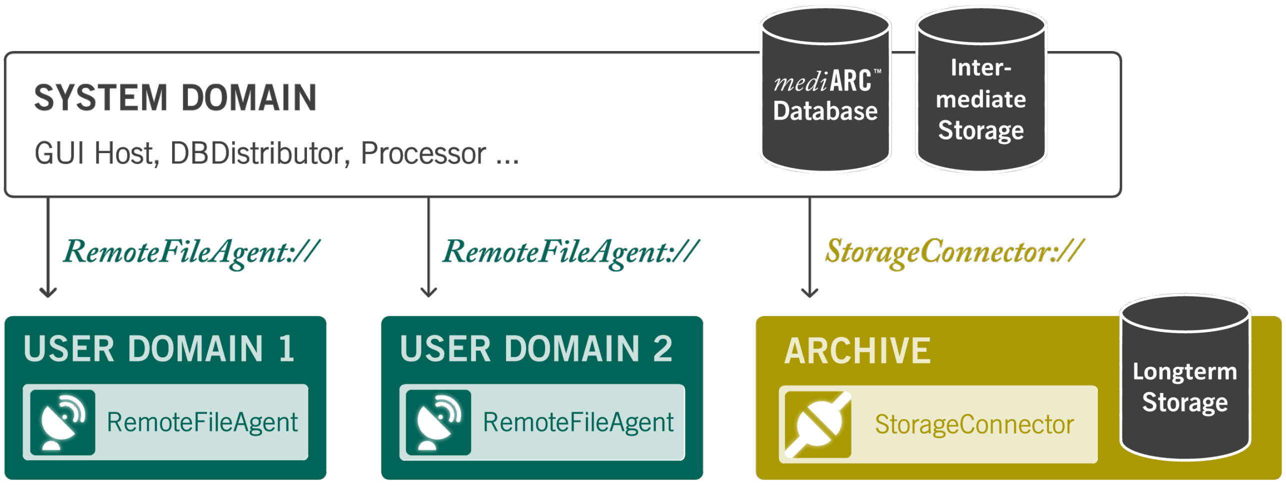
# 10 RemoteFileAgent (RFA)

## 10.1 Overview

RemoteFileAgent is used on target side by FileManager to securely transfer files from the system to the user in his User Domain and to control the file access rights of the requesting user within an Active Directory.

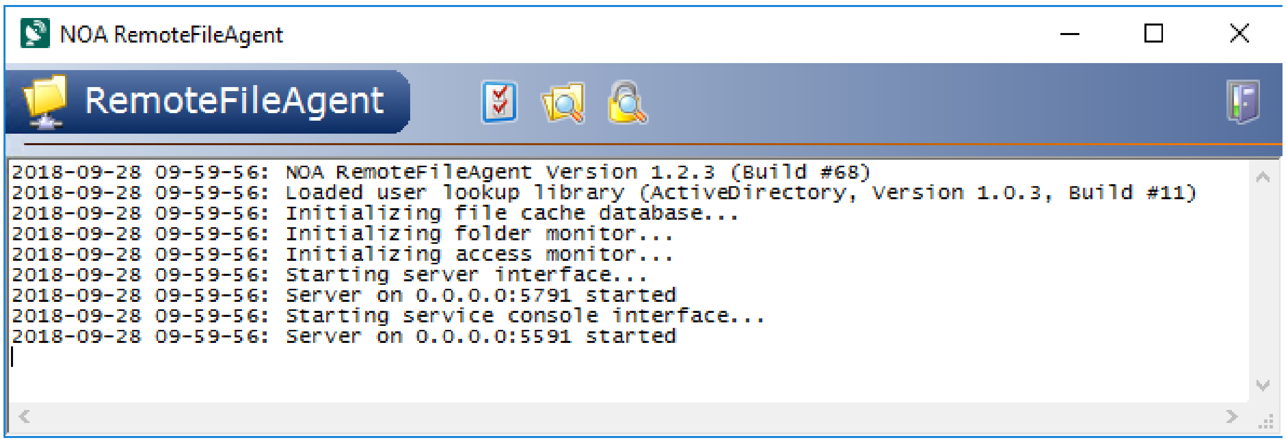
*If desired the RemoteFileAgent can monitor the Prelisten Cache and/or the users private paths and delete old files if necessary.*

In a mediARC environment the RemoteFileAgent needs to run on a Windows server in a User Access Domain, usually in combination with the WebPreviewServer (to manage the Prelisten Cache) or other mediARC processors.



*Access Domains Of mediARC*

Transfer of files is managed over NoaComm which uses blocklevel based checksums to guarantee a safe delivery of even large files. Typically one RemoteFileAgent per User Domain is used. A user access domain can be a local network within the premises of the running mediARC system, or a distant network, such as a regional station or a mobile laptop.



```
2018-09-28 09-59-56: NOA RemoteFileAgent Version 1.2.3 (Build #68)
2018-09-28 09-59-56: Loaded user lookup library (ActiveDirectory, Version 1.0.3, Build #11)
2018-09-28 09-59-56: Initializing file cache database...
2018-09-28 09-59-56: Initializing folder monitor...
2018-09-28 09-59-56: Initializing access monitor...
2018-09-28 09-59-56: Starting server interface...
2018-09-28 09-59-56: Server on 0.0.0.0:5791 started
2018-09-28 09-59-56: Starting service console interface...
2018-09-28 09-59-56: Server on 0.0.0.0:5591 started
```

*mediARC RemoteFileAgent*

Following buttons are available:

Change Ini Data: Shows the application wide settings window.

Show Folder Monitor: Shows the folder monitor window.

Show Access Monitor: Shows the access monitor window.

*Note that under normal conditions, RemoteFileAgent will be executed as a Windows service. On occasions, it is possible to run RemoteFileAgent as an executable program. Please make sure to stop the service before running the executable.*

*Note that RemoteFileAgent does not have a LicenseServer connection, which is why it can be turned on and off anytime without influencing other mediARC modules.*

## 10.2 Installation

To install or reinstall the RemoteFileAgent, simply run the installer and follow the instructions. During installation you can decide whether you want to install RemoteFileAgent as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

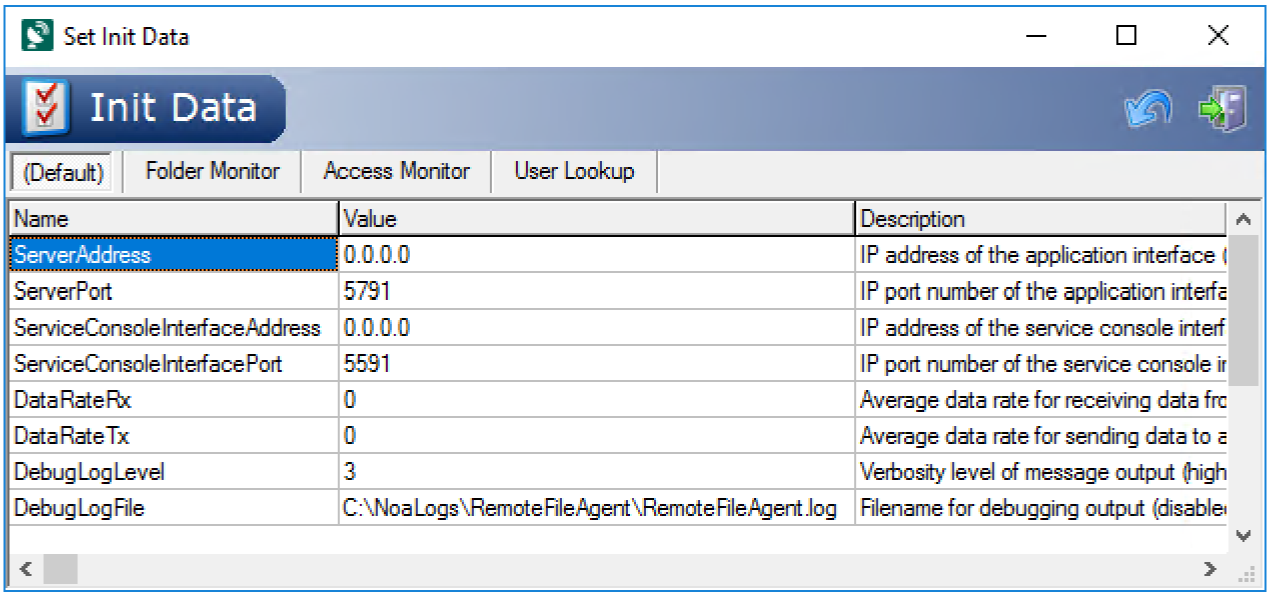
## 10.3 Configuration

*The configuration data of RemoteFileAgent is stored in the Windows registry, thus it is available for the service as well as for the application. Please take care which Windows user account you use when configuring the application as the settings will be available only for this user account. This is important when running the service version later on*

> *because the service must then be configured to use the same user account which was used during the application configuration step.*
>
> *The registry entries can be found in HKEY_CURRENT_USER\SOFTWARE\NOA\RemoteFileAgent.*

### 10.3.1 Init Data Window

There are some application-wide settings that can be found either using the Change Ini Data button or remotely by using the ServiceConsole.



| Name | Value | Description |
|---|---|---|
| ServerAddress | 0.0.0.0 | IP address of the application interface ( |
| ServerPort | 5791 | IP port number of the application interfa |
| ServiceConsoleInterfaceAddress | 0.0.0.0 | IP address of the service console interf |
| ServiceConsoleInterfacePort | 5591 | IP port number of the service console ir |
| DataRateRx | 0 | Average data rate for receiving data frc |
| DataRateTx | 0 | Average data rate for sending data to a |
| DebugLogLevel | 3 | Verbosity level of message output (high |
| DebugLogFile | C:\NoaLogs\RemoteFileAgent\RemoteFileAgent.log | Filename for debugging output (disable |

*RemoteFileAgent Init Data Window*

The following Init Data settings are used by the RemoteFileAgent:

**Default Tab**
ServerAddress: The IP interface address for listening to connections (may be 0.0.0.0 to listen on any interface). Defaults to "0.0.0.0".
ServerPort: The IP interface port number for listening to connections. Defaults to "5791".
ServiceConsoleInterfaceAddress: The IP interface address for listening for ServiceConsole connections (may be 0.0.0.0 to listen on any interface). Defaults to "0.0.0.0".
ServiceConsoleInterfacePort: The IP interface port number for listening for ServiceConsole connections. Defaults to "5591".
DataRateRx: The average data rate for receiving data from a remote location. Disabled if rate is 0.
DataRateTx: The average data rate for sending data to a remote location. Disabled if rate is 0.
DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\RFAlogs\RFA.log"). If a file name is provided, debug messages will be written to this file. The

amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

## Folder Monitor Tab

*Note that the RemoteFileAgent receives its files depending on the folder settings which are set up in the mediARC file access dialog (Administration | File Access in mediARC GUI), and the local settings for prelisten directories are only used to monitor the size of the directories!*

PrelistenDirs: Comma-separated list of directories which shall be scanned for file size and age limits. These directories are not filled by the FileManager, but only monitored by the RemoteFileAgent.
PrelistenMaxFileAge: Number of days for which the files should be preserved. Files older than the defined number of days will be deleted by the RemoteFileAgent. Defaults to "0".
PrelistenMaxDirSize: Maximum of disk usage (in megabytes). Defaults to "0".
PrelistenMinDiskFree: Minimum left free disk space (in megabytes). Defaults to "0".
PrivateDirs: Comma-separated list of directories which shall be scanned for file size and age limits. These directories are not filled by FileManager, but only monitored by RemoteFileAgent.
PrivateMaxFileAge: Number of days for which the files should be preserved. Files older than the defined number of days will be deleted by the RemoteFileAgent. Defaults to "0".
PrivateMaxDirSize: Maximum of disk usage (in megabytes). Defaults to "0".
PrivateMinDiskFree: Minimum left free disk space (in megabytes). Defaults to "0".

## Access Dialog Tab

EnableAccessUpdate: Enables automatic update of file access rights.
DefaultAccessTimeout: The default value for providing access rights (in hours). Defaults to "1".

**User Lookup Tab**
EnableUserLookup: Enables response to user lookup requests as used by the ProcessorHost user synchronization. For more information see the ProcessorHost chapter User Synchronization.
UseHashedSid: Forces user lookup to hash SID values and use an internal hash mapping database.
GroupNamePrefix: Prefix used for mediARC groups within Active Directory (may be quoted if containing spaces).
UserCategoryPrefix: Prefix used for mediARC user categories within Active Directory (may be quoted if containing spaces)
IncludeDomainPrefix: Include domain name as a prefix for returned user names.

### 10.3.2 mediARC File Access

Note that Access Methods and Paths that are required by the FileManager need to be defined in mediARC GUI using the menu Administration | File Access.

*For further information about File Access configuration in mediARC, please refer to the mediARC Domains TechNote or the mediARC User Configuration and Rights Management Administrator Manual.*

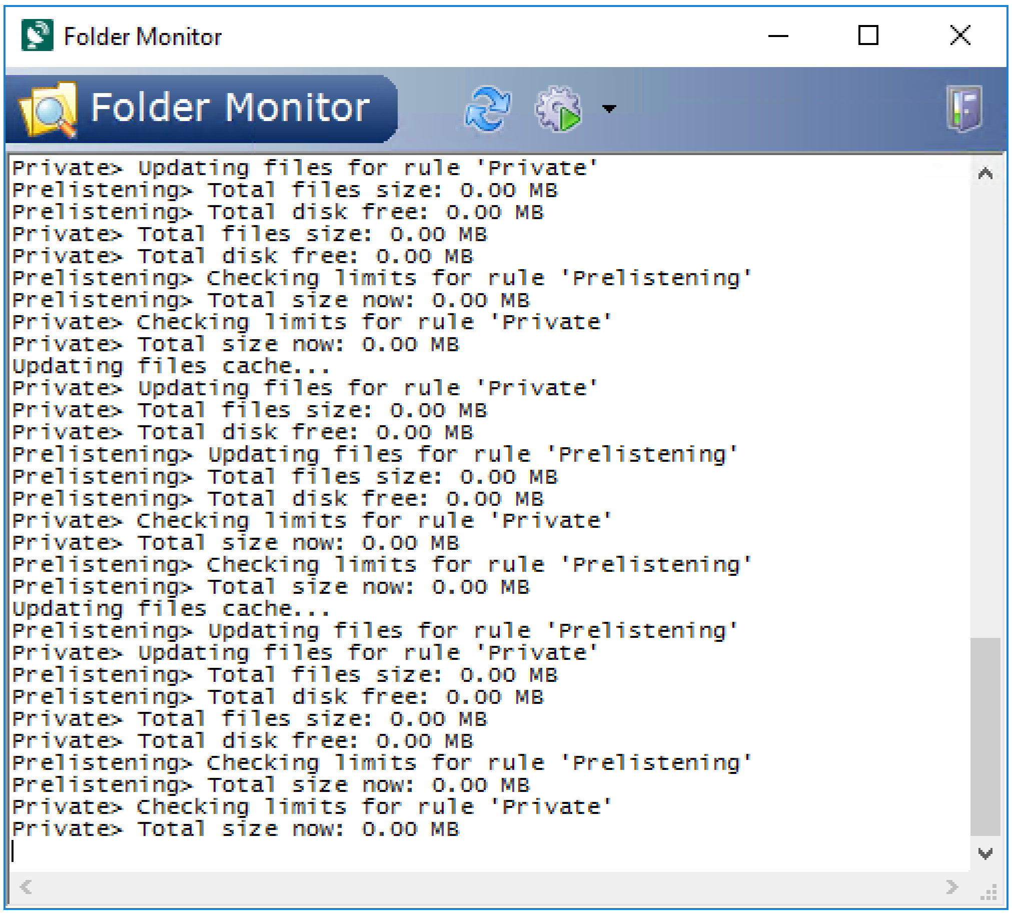
### 10.3.3 Folder Monitor

If desired the RemoteFileAgent can monitor the Prelisten Cache and/or the users' Private Folders and delete old files if necessary. Each of these directory lists can use three triggers to delete unused files:

- MaxFileAge: deletes files that are older than the specified number of days.
- MaxDirSize: deletes the oldest files until the total size of files in the directory list is below the size limit.
- MinDiskFree: deletes the oldest files until the free disk space of the drives in the directory list is above the limit.

*Warning: if a trigger is not used it must be set to zero! By default no triggers are used.*

The directories to monitor are set by the 'PrelistenDirs' and 'PrivateDirs' configuration properties in the RemoteFileAgent's Init Data Window.



```
Private> Updating files for rule 'Private'
Prelistening> Total files size: 0.00 MB
Prelistening> Total disk free: 0.00 MB
Private> Total files size: 0.00 MB
Private> Total disk free: 0.00 MB
Prelistening> Checking limits for rule 'Prelistening'
Prelistening> Total size now: 0.00 MB
Private> Checking limits for rule 'Private'
Private> Total size now: 0.00 MB
Updating files cache...
Private> Updating files for rule 'Private'
Private> Total files size: 0.00 MB
Private> Total disk free: 0.00 MB
Prelistening> Updating files for rule 'Prelistening'
Prelistening> Total files size: 0.00 MB
Prelistening> Total disk free: 0.00 MB
Private> Checking limits for rule 'Private'
Private> Total size now: 0.00 MB
Prelistening> Checking limits for rule 'Prelistening'
Prelistening> Total size now: 0.00 MB
Updating files cache...
Prelistening> Updating files for rule 'Prelistening'
Private> Updating files for rule 'Private'
Prelistening> Total files size: 0.00 MB
Prelistening> Total disk free: 0.00 MB
Private> Total files size: 0.00 MB
Private> Total disk free: 0.00 MB
Prelistening> Checking limits for rule 'Prelistening'
Prelistening> Total size now: 0.00 MB
Private> Checking limits for rule 'Private'
Private> Total size now: 0.00 MB
```

*RemoteFileAgent Folder Monitor Window*

The following buttons are available:

Refresh File List: Refreshes the current view.

Check Folder Limits: Runs the check immediately instead of waiting for the next scheduled run cycle.

The drop down menu of the Check Folder Limits button additionally offers following options:
Check Now: Runs the check immediately instead of waiting for the next scheduled run cycle. The default action (bold) that is triggered when the button is clicked.
Simulate Delete: When manually running a folder check, the oldest files in the Prelisten / Private folders will be deleted depending on the rules specified in the Init Data settings. When "Simulate Delete" is enabled, the affected files are just listed instead of actually removed from disk.
Disable Timer: Toggles on/off the timer that runs the checks in cycles on automatic schedule.

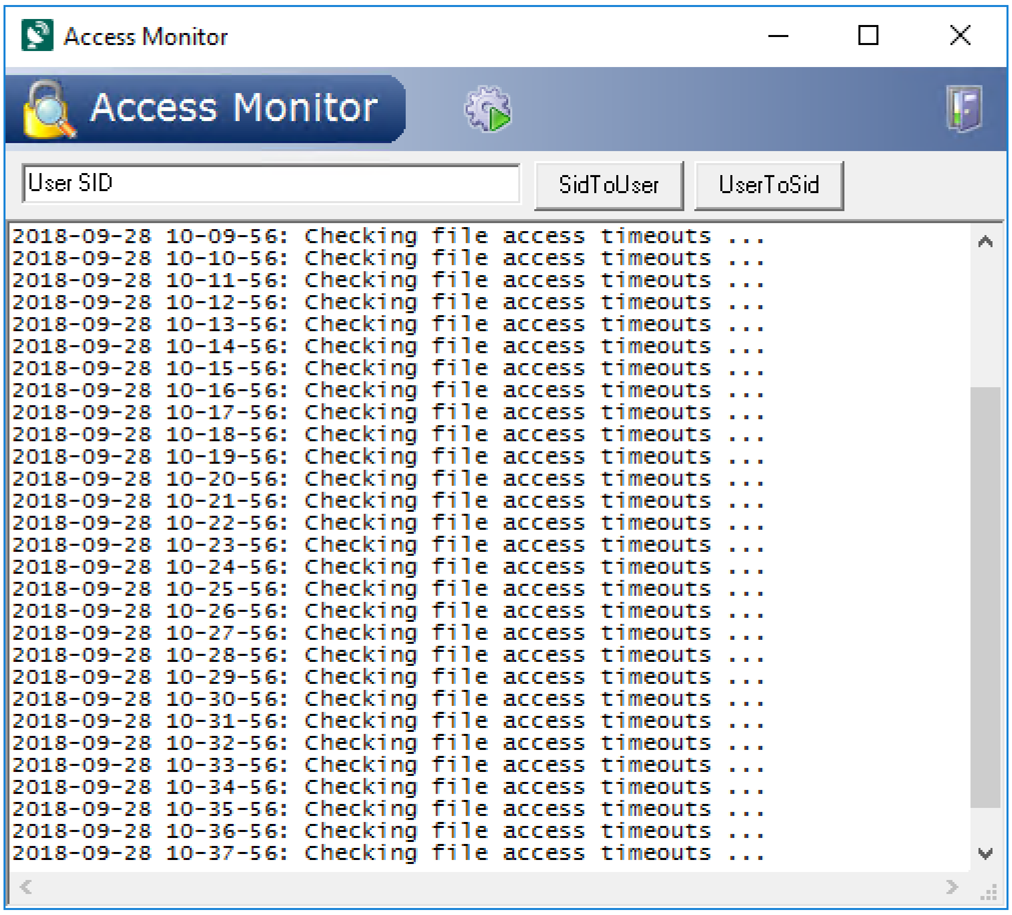
## 10.3.4 Access Monitor

The Access Monitor window allows to monitor file access rights assigned to files in the Prelisten Cache, for debugging purposes.

Whenever a user clicks on the prelisten / preview button in the mediARC GUI, the application tries to read the file from the Prelisten Cache which is located in the User Domain of the requesting user. If the file is not online, the Prelisten Flow is used by mediARC to provide the file from the source (the mediARC GUI does not read the audio file directly from the archive as it will typically not have direct access to the archive).

In the last step of the Prelisten Flow, FileManager will read the file from the archive and pass it to the RemoteFileAgent running in the local network of the user that triggered the prelisten request. The RemoteFileAgent retrieves the file from the FileManager and stores it to the Prelisten Cache. Finally it ensures that the requesting user has proper access rights to read the file (as users do not have access to all of the files in the Prelisten Cache but only to the ones they have recently requested).

The RemoteFileAgent will remove the rights after the timeout defined in the DefaultAccessTimeout in the Folder Monitor tab of the RFA's Init Data settings.



```
2018-09-28 10-09-56: Checking file access timeouts ...
2018-09-28 10-10-56: Checking file access timeouts ...
2018-09-28 10-11-56: Checking file access timeouts ...
2018-09-28 10-12-56: Checking file access timeouts ...
2018-09-28 10-13-56: Checking file access timeouts ...
2018-09-28 10-14-56: Checking file access timeouts ...
2018-09-28 10-15-56: Checking file access timeouts ...
2018-09-28 10-16-56: Checking file access timeouts ...
2018-09-28 10-17-56: Checking file access timeouts ...
2018-09-28 10-18-56: Checking file access timeouts ...
2018-09-28 10-19-56: Checking file access timeouts ...
2018-09-28 10-20-56: Checking file access timeouts ...
2018-09-28 10-21-56: Checking file access timeouts ...
2018-09-28 10-22-56: Checking file access timeouts ...
2018-09-28 10-23-56: Checking file access timeouts ...
2018-09-28 10-24-56: Checking file access timeouts ...
2018-09-28 10-25-56: Checking file access timeouts ...
2018-09-28 10-26-56: Checking file access timeouts ...
2018-09-28 10-27-56: Checking file access timeouts ...
2018-09-28 10-28-56: Checking file access timeouts ...
2018-09-28 10-29-56: Checking file access timeouts ...
2018-09-28 10-30-56: Checking file access timeouts ...
2018-09-28 10-31-56: Checking file access timeouts ...
2018-09-28 10-32-56: Checking file access timeouts ...
2018-09-28 10-33-56: Checking file access timeouts ...
2018-09-28 10-34-56: Checking file access timeouts ...
2018-09-28 10-35-56: Checking file access timeouts ...
2018-09-28 10-36-56: Checking file access timeouts ...
2018-09-28 10-37-56: Checking file access timeouts ...
```

*RemoteFileAgent Access Monitor Window*

The following buttons are available:

Check Access Timeouts: Runs the check for expired access rights immediately instead of waiting for the next scheduled run cycle.

A simple user name / SID translator is available to help the debugging process:

- SidToUser: translates the given SID ID to a user name.
- UserToSid: translates the given user name to a SID ID.

# WebPreviewServer (WPS)

# 11 WebPreviewServer (WPS)

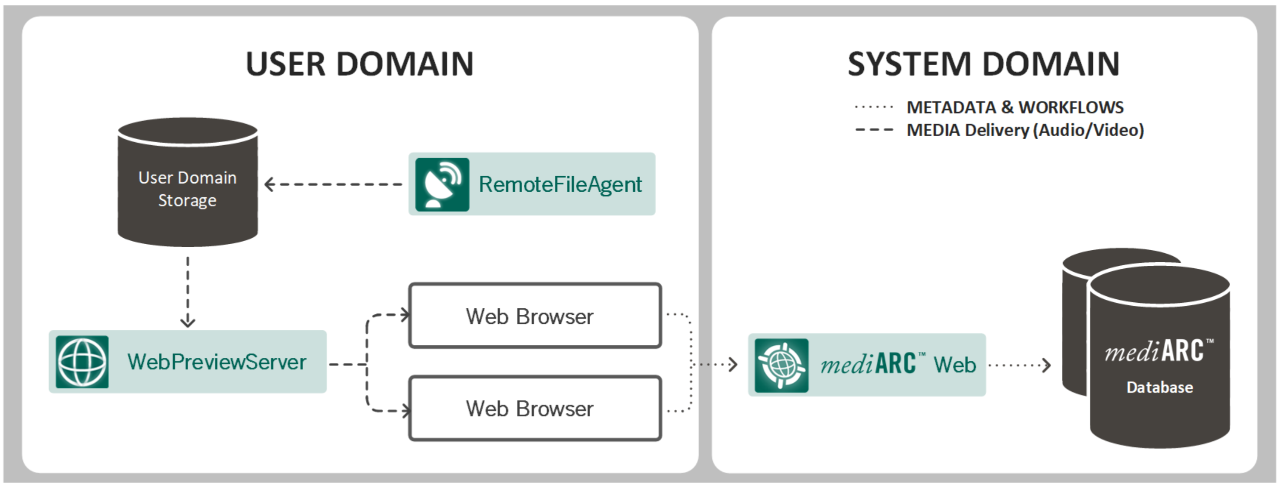
## 11.1 Overview

WebPreviewServer acts as a streaming server for prelisten/preview requests from the mediARC WEB client application. It serves media files to the player on the client machine using the HTTP protocol for maximum compatibility. WebPreviewServer uses the Prelisten Cache of the mediARC system for reading media files.

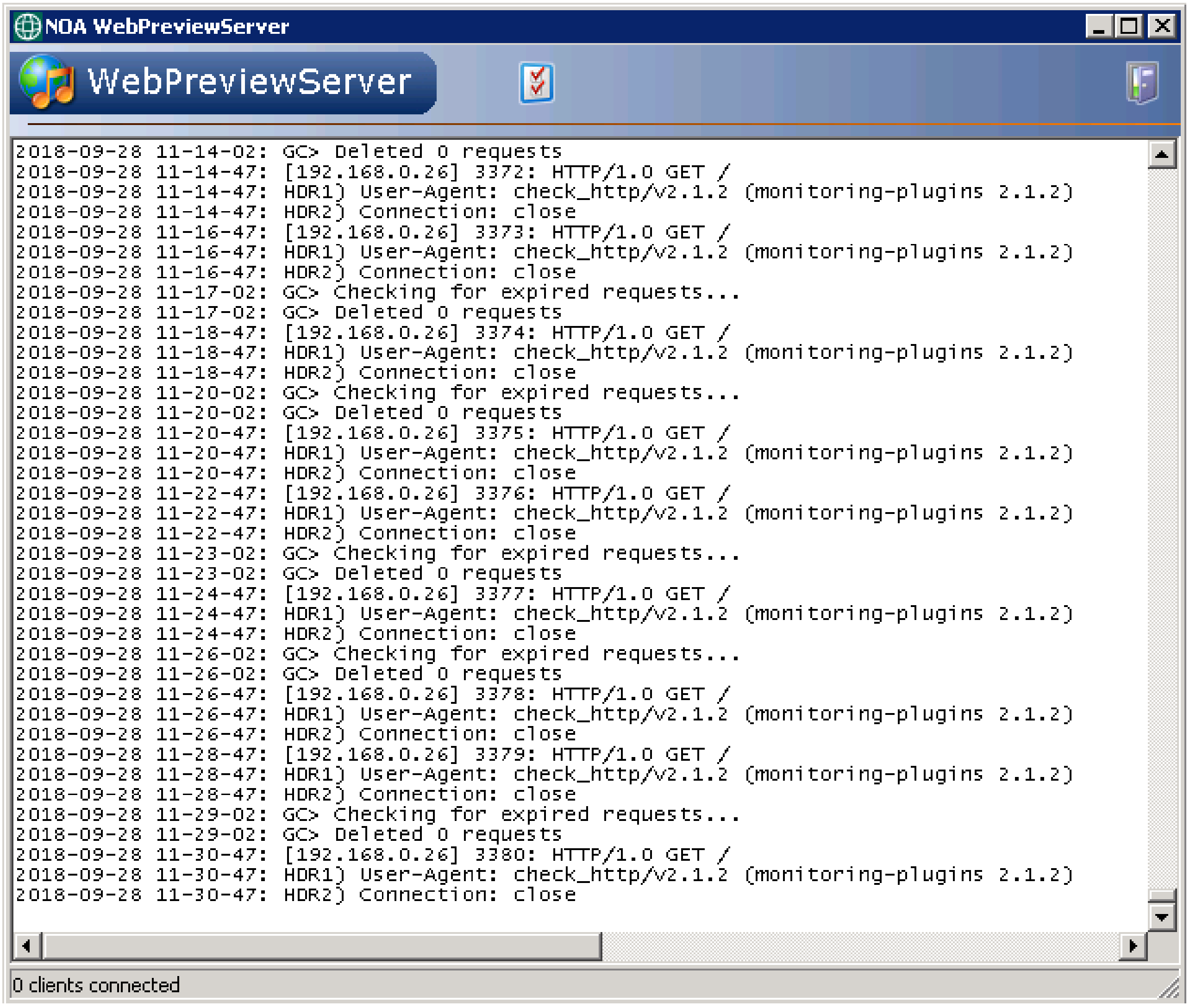
In a mediARC environment the WebPreviewServer is usually located on the same machine as RemoteFileAgent, which manages the Prelisten Cache (if an Oracle Client is available there).

The base URL is `http://<server>:<port>/prelisten/`
`e.g. http://noa-wps:5799/prelisten)`

*WebPreviewServer restricts access to media data for prelistening/previewing by limiting the playable file range and the time the file URL is available. The application also automatically adapts the download bitrate of media files to allow a maximum of concurrent users.*



*WebPreviewServer used for Web Preview*



```
2018-09-28 11-14-02: GC> Deleted 0 requests
2018-09-28 11-14-47: [192.168.0.26] 3372: HTTP/1.0 GET /
2018-09-28 11-14-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-14-47: HDR2) Connection: close
2018-09-28 11-16-47: [192.168.0.26] 3373: HTTP/1.0 GET /
2018-09-28 11-16-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-16-47: HDR2) Connection: close
2018-09-28 11-17-02: GC> Checking for expired requests...
2018-09-28 11-17-02: GC> Deleted 0 requests
2018-09-28 11-18-47: [192.168.0.26] 3374: HTTP/1.0 GET /
2018-09-28 11-18-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-18-47: HDR2) Connection: close
2018-09-28 11-20-02: GC> Checking for expired requests...
2018-09-28 11-20-02: GC> Deleted 0 requests
2018-09-28 11-20-47: [192.168.0.26] 3375: HTTP/1.0 GET /
2018-09-28 11-20-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-20-47: HDR2) Connection: close
2018-09-28 11-22-47: [192.168.0.26] 3376: HTTP/1.0 GET /
2018-09-28 11-22-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-22-47: HDR2) Connection: close
2018-09-28 11-23-02: GC> Checking for expired requests...
2018-09-28 11-23-02: GC> Deleted 0 requests
2018-09-28 11-24-47: [192.168.0.26] 3377: HTTP/1.0 GET /
2018-09-28 11-24-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-24-47: HDR2) Connection: close
2018-09-28 11-26-02: GC> Checking for expired requests...
2018-09-28 11-26-02: GC> Deleted 0 requests
2018-09-28 11-26-47: [192.168.0.26] 3378: HTTP/1.0 GET /
2018-09-28 11-26-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-26-47: HDR2) Connection: close
2018-09-28 11-28-47: [192.168.0.26] 3379: HTTP/1.0 GET /
2018-09-28 11-28-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-28-47: HDR2) Connection: close
2018-09-28 11-29-02: GC> Checking for expired requests...
2018-09-28 11-29-02: GC> Deleted 0 requests
2018-09-28 11-30-47: [192.168.0.26] 3380: HTTP/1.0 GET /
2018-09-28 11-30-47: HDR1) User-Agent: check_http/v2.1.2 (monitoring-plugins 2.1.2)
2018-09-28 11-30-47: HDR2) Connection: close
```

0 clients connected

*mediARC WebPreviewServer Window*

The following buttons are available:

Change Init-Data: Shows the application wide settings window.

## 11.2 Installation

To install or reinstall the WebPreviewServer, simply run the installer and follow the instructions. During installation you can decide whether you want to install WebPreviewServer as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

**Prerequisites**

- WebPreviewServer needs a pre-installed OracleClient (32-bit version) pointing to the right database schema.

- Upon the first start of the application at least the database connection settings need to be configured (TNS Alias, username, password).

*Note that in 64-bit versions of Windows, the* WebPreviewServer *needs to be installed in a directory whose full path does not contain special characters (e.g. C:\NOA\) and not in the default directory "C:\Program Files (x86)\NOA\" because the brackets "(" and ")" will break the connection!*

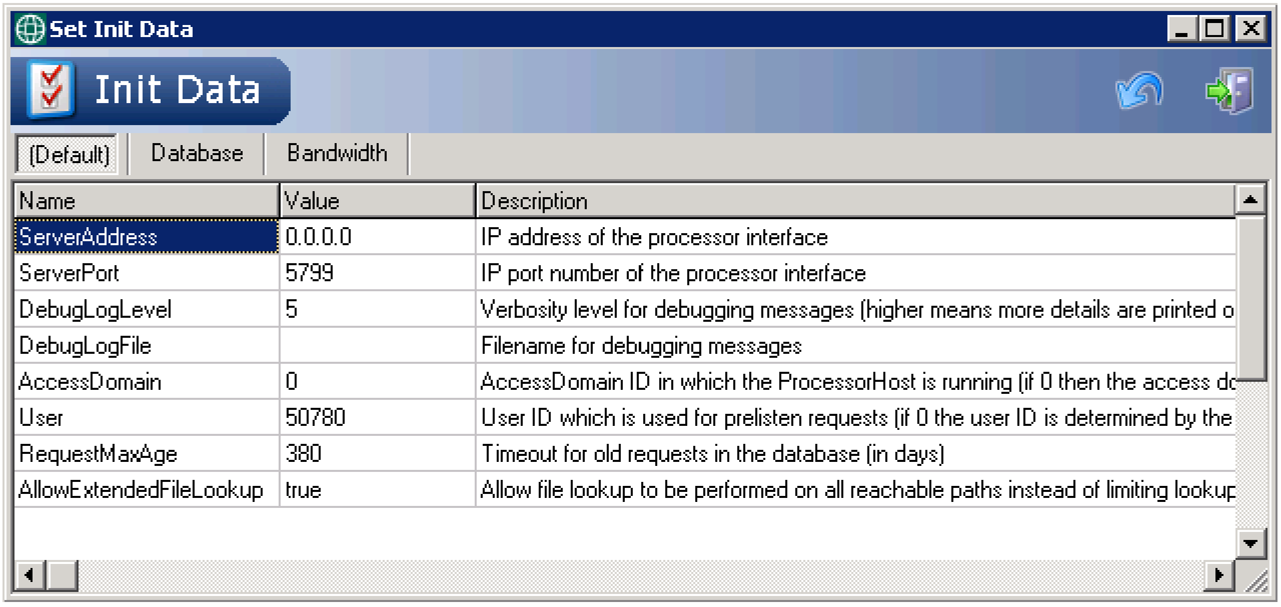
## 11.3 Configuration

*The configuration data of WebPreviewServer is stored in the Windows registry, thus it is available for the service as well as for the application. Please take care which Windows user account you use when configuring the application as the settings will be available only for this user account. This is important when running the service version later on because the service must then be configured to use the same user account which was used during the application configuration step.*

*The registry entries can be found in HKEY_CURRENT_USER\SOFTWARE\NOA\WebPreviewServer.*

### 11.3.1 Init Data Window

There are some application-wide settings that can be found either using the Change Ini Data button or remotely by using the ServiceConsole.



| Name | Value | Description |
|---|---|---|
| ServerAddress | 0.0.0.0 | IP address of the processor interface |
| ServerPort | 5799 | IP port number of the processor interface |
| DebugLogLevel | 5 | Verbosity level for debugging messages (higher means more details are printed o |
| DebugLogFile | | Filename for debugging messages |
| AccessDomain | 0 | AccessDomain ID in which the ProcessorHost is running (if 0 then the access dc |
| User | 50780 | User ID which is used for prelisten requests (if 0 the user ID is determined by the |
| RequestMaxAge | 380 | Timeout for old requests in the database (in days) |
| AllowExtendedFileLookup | true | Allow file lookup to be performed on all reachable paths instead of limiting lookup |

*WebPreviewServer Init Data Window*

The following Init Data settings are used by the WebPreviewServer:

**Defaul Tab**
ServerAddress: The IP interface address for listening for prelisten request connections. The special value "0.0.0.0" may be used to listen on all available interfaces. Defaults to "0.0.0.0".
ServerPort: The IP interface port number for listening for prelisten request connections. Defaults to "5799".
DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\WPSlogs\WPS.log"). If a file name is provided, debug messages will be written to this file. The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

AccessDomain: The Access Domain ID in which WebPreviewServer is running, to determine the Prelisten Cache. If "0" (the default) then the Access Domain of the User ID is used.
User: User ID which is used for prelisten requests. If "0" (the default) the user ID is determined by the Windows user running the application.
RequestMaxAge: The timeout for old prelisten requests in the database (in days). Defaults to "120" days.
AllowExtendedFileLookup: Allows file lookup to be performed on all reachable paths instead of limiting lookup to the Prelisten Cache (might slow down the server depending on throughput and latency of those paths). Defaults to "false".

**Database Tab**
DBServer: The TNS Alias of the Oracle database connection.
DBUsername: Login credentials for the database server.
DBPassword: Login credentials for the database server.

**Bandwidth Tab**

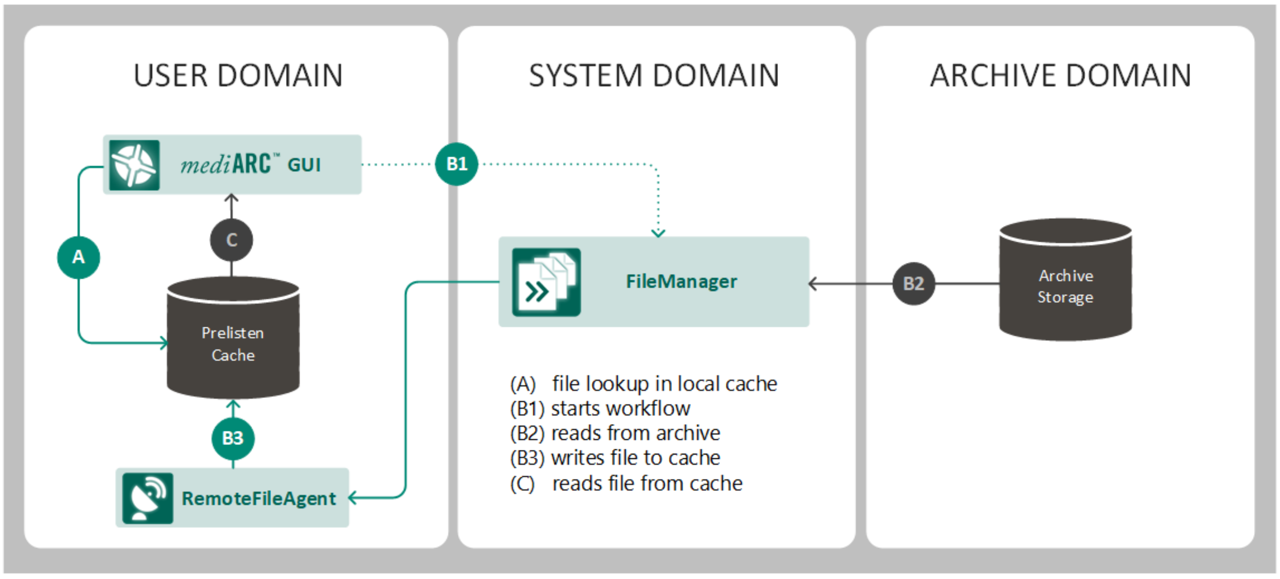
BandwidthBitrateScale: The scale factor for determining the media streaming rate. The streaming rate is calculated by multiplying the actual file bitrate with the scale factor. Useful values for the scale factor are values between 1.2 and 5.0. The default value "0" employs a hard-coded scale > 1, allowing the client to maintain buffers with low burst rates.
BandwidthMinRate: The minimum rate for media streaming in kilobytes per second. If set to "0", a default value of 32 kbps is used.
BandwidthMaxRate: The maximum rate for media streaming in kilobytes per second. The default value "0" provides unlimited bandwidth.

*Note that you can also change the bandwidth settings in the Windows registry: HKEY_CURRENT_USER\Software\NOA\WebPreviewServer\bandwidth*
All settings are efficient per stream. *Nominal yields given by BandwidthBitrateScale may be limited by BandwidthMinRate and BandwidthMaxRate at the upper/lower threshold.*

## 11.3.2 Prelisten Cache

WebPreviewServer uses the Prelisten Cache for reading in media files. In a basic setup, it is recommended that the WebPreviewServer is run under the Windows account for NOA processor applications (i.e. "noa-auto") and that a Prelisten Cache is configured for the User Domain (see Administration | File Access in mediARC GUI).

If a file is not available in the Prelisten Cache, a Prelisten Flow is started.



For details on the prelisten mechanism have a look at Prelisten Flow.

**Database Configuration**

The Oracle stored procedure/function InitWebPrelisten() is responsible for passing prelisten/preview requests from mediARC WEB clients to the WebPreviewServer. Current versions of InitWebPrelisten() function look-up the matching WebPreviewServer using the defined access methods between user domain and file archive domain. For example, to configure WebPreviewServer on host "noa-wps" for providing prelisten files stored in the archive domain "Archive" to users of the "Web Users" access domain, one will define an access method with the value "WebPreviewServer://noa-wps/" on "Web Users".

The InitWebPrelisten() function also contains a base URL which is used if no matching access method could be found. This URL string should be configured upon a first-time installation on Oracle database side. It will use the hostname of the machine WebPreviewServer is installed on and the port for prelisten requests as specified above to form the base URL:
http://<server>:<port>/prelisten/
The base URL for the upper example with WebPreviewServer running on host "noa-wps" is: http://noa-wps:5799/prelisten/.

*Note that older versions of the InitWebPrelisten() function do not support configuration using Access Methods and it is necessary to define the base URL manually when adding/changing the WebPreviewServer hosts.*

*For further information, please refer to the mediARC Workflows documentation.*

## 11.4 Status Information

The WebPreviewServer allows mediARC administrators to monitor the status of the application through a special "admin page" available on the following URL:

http://<server>:<port>/admin/

On the machine where the WebPreviewServer application is installed this URL resolves to "http://localhost:5799/admin/" (be careful to include the last "/").

*Access to the admin status page is granted to all mediARC users who own the "Flow.Admin" key.*

The following information is available in the admin menu:
App Info: The general information about the WebPreviewServer, including Version, Startup Time, Database Connection Information, Prelisten Cache Paths etc.

I/O Info: The general information about hardware and network resources, incl. CPU, Memory, Client Count, Throughput etc.
Prelisten Req List: A comprehensive list of all prelisten requests incl. File Name, Media Source Name, Request Date, Client Address etc.
Attachment Req List: A comprehensive list of all attachment requests incl. File Name, Request Date, Client Address etc.
Client List: The list of all currently connected clients.

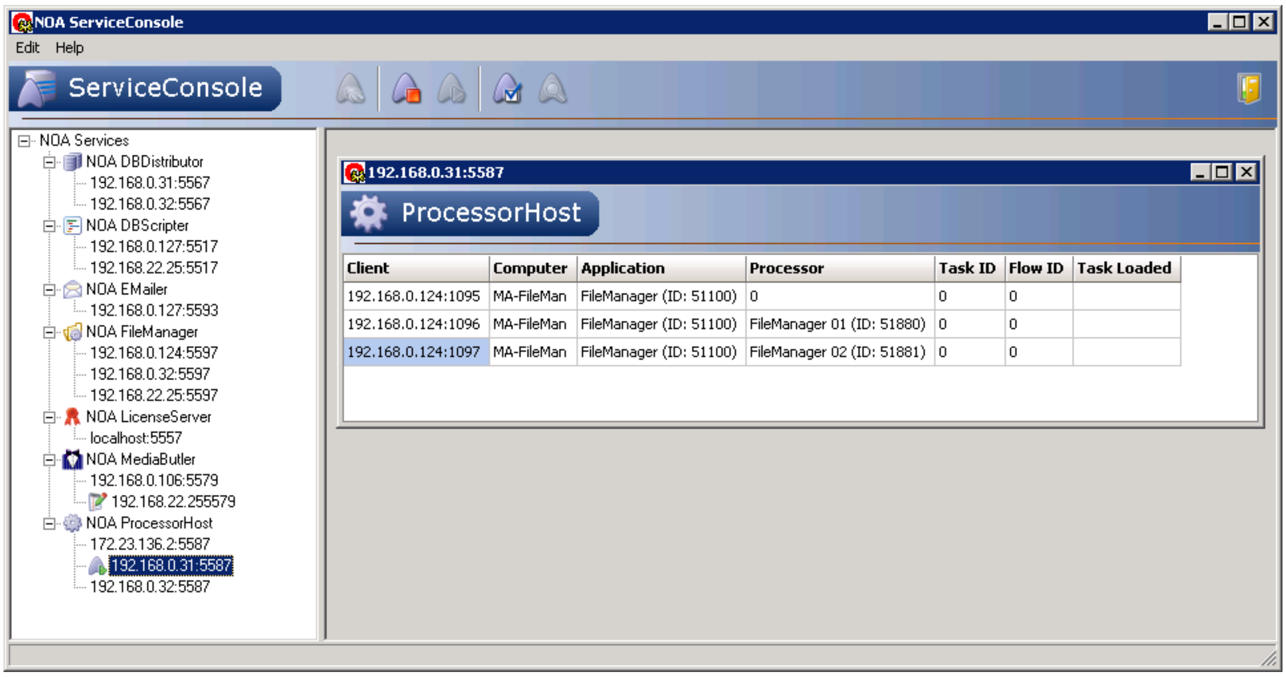
# ServiceConsole (SC)

# 12 ServiceConsole (SC)

## 12.1 Overview

The ServiceConsole is part of the actLINE product family and it can be used to pause, resume, monitor and configure nearly all NOA service applications.

By default, the ServiceConsole will query the LicenseServer for active Processor Services and display the instance in the connection tree, including the LicenseServer itself. Manual configuration is only required for Services that have no connection to the LicenseServer (e.g. RemoteFileAgent). See list of supported Services below. Another case where connections need to be set manually is if the ServiceConsole is running in a different network than the LicenseServer.



| Client | Computer | Application | Processor | Task ID | Flow ID | Task Loaded |
|---|---|---|---|---|---|---|
| 192.168.0.124:1095 | MA-FileMan | FileManager (ID: 51100) | 0 | 0 | 0 | |
| 192.168.0.124:1096 | MA-FileMan | FileManager (ID: 51100) | FileManager 01 (ID: 51880) | 0 | 0 | |
| 192.168.0.124:1097 | MA-FileMan | FileManager (ID: 51100) | FileManager 02 (ID: 51881) | 0 | 0 | |

*The ServiceConsole Main Window*

The following buttons are available:

Connect To Service: Connects to the currently selected Service.

Pause Service: The Service releases most of it's resources (like the ProcessorHost connection). The Service finishes it's currently loaded tasks before going into pause state so it might take some time until the pause state is entered.

Continue Service: The Service starts up and resumes normal operations.

Show Ini Dialog: A dialog is shown to configure the Init Data of the Service. Note that most of the settings will not take effect until the Service is paused/continued or restarted.

View Debug Log: If available, the debug log of the Service application is displayed.

The same functions are also available from the contextual menu after right-clicking a host in the NOA Services list.

The following menu items are available:

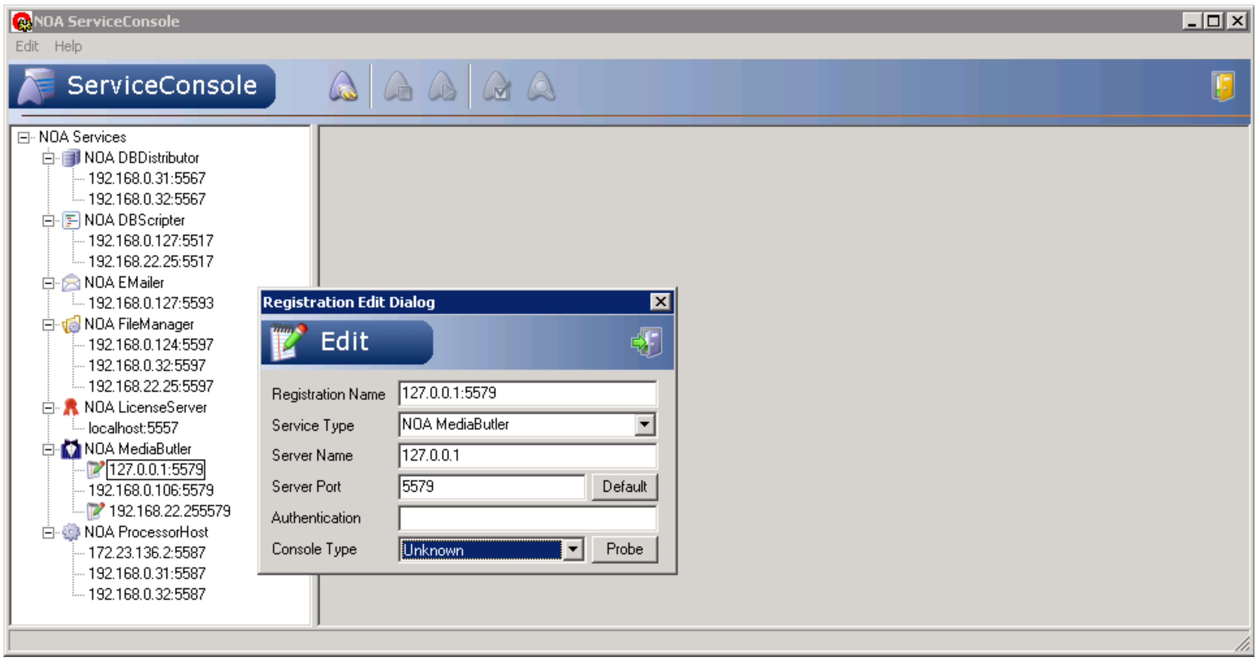
- Edit | Settings: Shows the application wide Settings window.
- Edit | Update Service List F5: Refreshes the list of available services.

The following Services are supported:

- Clip
- DBDistributor
- DBScripter
- Emailer
- FileCollector
- FileManager
- FolderScanner
- LicenseServer
- MediaButler
- ProcessorHost
- RemoteFileAgent
- StorageConnector
- WaveButler

### 12.1.1 Adding Services Manually

It is possible to add available Services in the NOA Services list manually. To do this, right-click on the appropriate service type icon or the 'NOA Services' root label in the tree view and choose Add Registration.



- NOA Services
  - NOA DBDistributor
    - 192.168.0.31:5567
    - 192.168.0.32:5567
  - NOA DBScripter
    - 192.168.0.127:5517
    - 192.168.22.25:5517
  - NOA EMailer
    - 192.168.0.127:5593
  - NOA FileManager
    - 192.168.0.124:5597
    - 192.168.0.32:5597
    - 192.168.22.25:5597
  - NOA LicenseServer
    - localhost:5557
  - NOA MediaButler
    - 127.0.0.1:5579
    - 192.168.0.106:5579
    - 192.168.22.255579
  - NOA ProcessorHost
    - 172.23.136.2:5587
    - 192.168.0.31:5587
    - 192.168.0.32:5587

| | |
|---|---|
| Registration Name | 127.0.0.1:5579 |
| Service Type | NOA MediaButler |
| Server Name | 127.0.0.1 |
| Server Port | 5579 |
| Authentication | |
| Console Type | Unknown |

*The Registration Edit Dialog Window*

A small Registration Edit Dialog window will appear where the required custom settings can be defined.

The following information needs to be provided:
- Registration Name: the name of the Service connection that will be displayed in the list.
- Service Type: the type of Service chosen from a dropdown list of supported Services.
- Server Name: the IP address of the ServiceConsole interface on which the Service is listening for connection requests.
- Server Port: the IP port of the ServiceConsole interface on which the Service is listening for connection requests (see chapter mediARC Network Ports).
  - use the Default button to reset the port number to the application's default value
- Authentication: If the current Windows user account is within the LicenseServer's Windows domain, the authentication field can be left empty. Else, enter an authentication string (see chapter User Authentication).
- Console Type: specifies whether the service application supports remote forms (e.g. the processor applications FolderScanner, FileCollector and WaveScanner) or not (most other applications). Usually it is a safe choice to leave the setting at "unknown" until experiencing problems.

### 12.1.2 Editing Services Manually

It is possible to change or delete available Services in the NOA Services list. To do this, select an existing Service connection in the list, right-click it and select:
- Edit Registration...: displays the Registration Edit Dialog window, which allows the user to edit:
  - the Registration Name for automatically added Service connections
  - the Registration Name, Service Type, Server Name, Server Port, Authentication and Console Type for manually added Service connections
- Delete Registration: deletes the selected Service connection from the list (no confirmation dialog).

## 12.2 Installation

To install or reinstall the ServiceConsole, simply run the installer and follow the instructions.
During installation you can decide whether you want to install the ServiceConsole as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

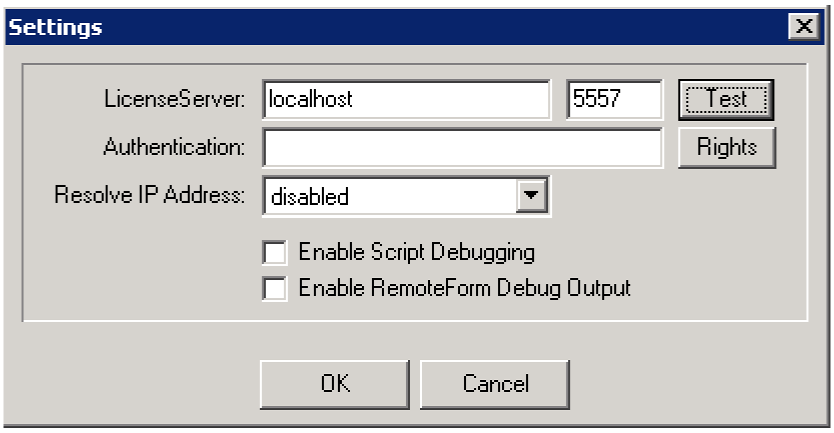
**Prerequisites**
- The LicenseServer needs to have the NTServer setting in the Init Data window configured.

## 12.3 Configuration

*The configuration data of the ServiceConsole is stored in the 'NoaServiceConsole.ini' text file in the ServiceConsole installation directory.*

### 12.3.1 Settings Window

There are some application-wide settings that can be found using the Edit | Settings... menu in the main window.



LicenseServer: localhost 5557
Authentication:
Resolve IP Address: disabled
☐ Enable Script Debugging
☐ Enable RemoteForm Debug Output

*The Settings Window of the ServiceConsole*

The following settings are used by the ServiceConsole:
LicenseServer: The Hostname / IP address and the Port number of the LicenseServer. Defaults to "localhost" and "5557" respectively.

- use the Test button to test the connection to the LicenseServer.

Authentication: The optional user authentication string used to get access rights from the LicenseServer. If empty, the local user account running the ServiceConsole will be used.

- use the Rights button to calculate effective rights of both the local user account running the ServiceConsole and the explicit user authentication string given.

Resolve IP Address: By default this setting is set to disabled and the machines running the Services are listed using only their IP addresses. Other options are:

- hostname: the IP is resolved to its <host name>.
- full qualified name: the IP is resolved to its <host name>.<domain name>

*Note: Depending on your network structure resolving hostnames might lead to application timeouts. If you encounter timeout problems upon application startup you should disable this configuration option.*

Enable Script Debugging: Shows a menu entry that displays messages from the procedure that loads an application window. These messages can be used by developers of remote applications to debug problematic behavior.

Enable RemoteForm Debug Output: Allows the developer of a remote form (see above) to verify what form data is received by the ServiceConsole. The ServiceConsole will write the form data into a file "Debug_RemoteForm.dfm" in the current folder.

### 12.3.2 User Authentication

The ServiceConsole does not use the default NOA user login and authentication method. Therefore, in theory, any user can run it.

However, when the ServiceConsole requests a connection to a mediARC service or an actLINE processor application, a certain level of user authentication does take place. This User Rights Management (URM) is handled by the LicenseServer, which is either pooling user account information from a Windows domain controller, or comparing locally defined user tokens (see chapter Explicit Users).

*For further information on how to configure user authentication for ServiceConsole users, see chapter LicenseServer > Configuration.*

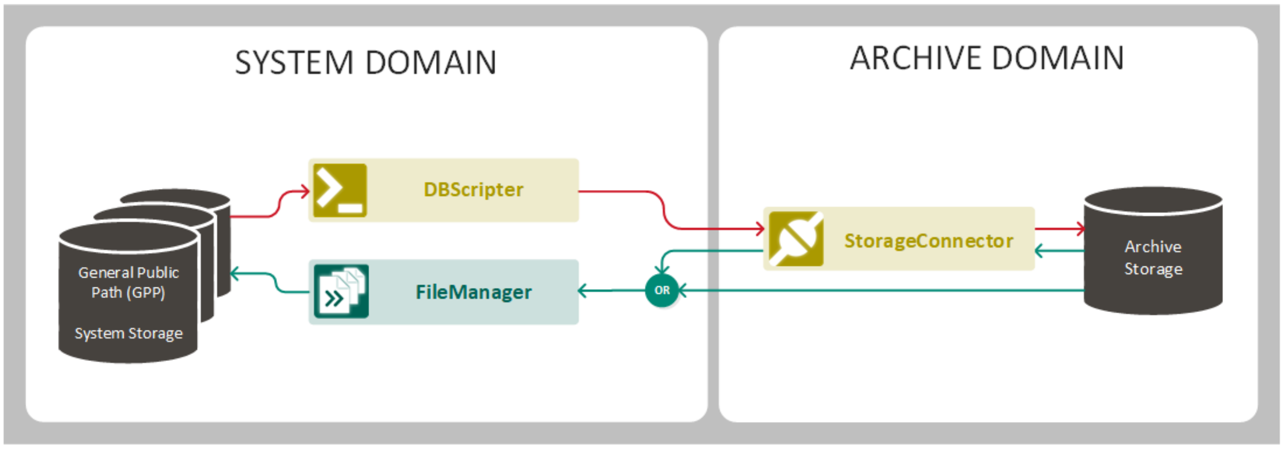
In ServiceConsole, go to the Edit | Settings... menu, and enter the defined user token in the Authentication field.

# StorageConnector (STORCON)

# 13 StorageConnector (STORCON)

## 13.1 Overview

The StorageConnector is part of the actLINE product family and it servers as a file access agent, which provides access to archive storage for the rest of the system. The purpose of the StorageConnector is acting as one point of entry for store or restore requests, isolating direct access from normal users and managing access rights.



*Typical use cases of the StorageConnector*

StorageConnector may be contacted by the FileManager or the 'UpdateStorageTask' of DBScripter. This method is using NoaComm protocol (NoaComm is a protocol definition that is used to exchange information between two computers. It's based on only two long-lasting and platform independent standards: TCP/IP and XML).

StorageConnector is able to run its own transactions on asynchronous protocols thus being enabled interacting with Hierarchical Storage Management (HSM) or Storage Content Management Systems (SCMS). Different file naming policies towards the storage can be implemented, in order to cope with specific demands from the storage system itself (most typically a limitation of maximum directory size in some NAS environments, or the maximum number of folders per node on some systems).

Currently there are two StorageConnector versions available:

- StorageConnector UNC: A generic interface to CIFS shares, typically used in radio archive environments or standard CIFS environments where a CIFS gateway exists.
- StorageConnector DIVA: A dedicated interface to ORACLE DIVA Archive, where Partial File Index & Restore for FFV1 is fully supported. This allows for much faster retrieve and access times when ordering short segments of video files. StorageConnector DIVA creates an index within mediARC derived from the NOA archival format FFV1 (which is created by MediaButler). This allows users to order segments from DIVA Tape Storage in a fraction of time, allowing for lower bandwidth on network and higher performance of tape archival and retrieval tasks. When using the StorageConnector DIVA, no additional PFR license is required for DIVA.

*Note that the StorageConnector is normally being run as a Windows service, and while there is a GUI version available, this is used only for debugging purposes (see chapter StorageConnector GUI).*

## 13.2 Installation

*Note that because StorageConnector is available in different versions, separate installers are also available. Please make sure you are using the right one.*
*The following chapters describe "StorageConnectorFile" as an example, but the same steps apply to other versions of StorageConnector as well.*

To install or reinstall the StorageConnector, simply run the installer and follow the instructions. During installation you can decide whether you want to install the StorageConnector as a service or as an application (or both).
After the installation is complete, have a look at the history file, which is installed with the application, for changes.

**Prerequisites**

- The machine needs to have network access to the archive storage.
- All running DBScripter instances should be turned off or made inactive via the ServiceConsole.

*Warning: When performing an update of an already running installation of StorageConnector, it is important to not overwrite the file TransDb.db3 (located in the StorageConnector's installation folder), as it contains information about currently running/scheduled transactions!*

## 13.3 Configuration

*The configuration data of the StorageConnector is stored in:*

- *the 'Policy.ini' text file in the directory selected during the installation process (by default C:\Noa\StorageConnector\PolicyFile\Policy.ini)*
- *the Windows registry under the following paths:*
  - *32-bit Windows: HKEY_LOCAL_MACHINE\SOFTWARE\NOA\StorageConnector\*
  - *64-bit Windows: HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432node\NOA\StorageConnector\*

*Note that further configuration is also necessary in mediARC GUI, menu Administration | File Access.*

## 13.3.1 Policy File

Before running the StorageConnector, the Policy File 'Policy.ini' needs to be configured manually, in a text editor (e.g. notepad.exe).

> *Warning: If no 'Policy.ini' file is available, the StorageConnector application or service will not start!*

The file is usually located in the directory selected during the installation process. The default path is '*C:\Noa\StorageConnector\PolicyFile\Policy.ini*'. The installer supplies also a 'Policy-Example.ini' file in the same directory, to make the initial configuration much simpler (just edit and save as / rename the file to 'Policy.ini').

> *Note that the syntax of the Policy File is different for each version of StorageConnector (like StorageConnectorFile.exe or StorageConnectorDiva.exe)!*

**StorageConnector UNC:**

```
[Policy:1]
; base path for storing files with policy "1"
Path1.Path=\\Server\mediARC\Archive\BWF\

[Policy:2]
; base path for storing files with policy "2", with a local Temp directory defined
Path1.Path=\\Server\MediARC\Archive\MP3\
Path1.LocalTmpPath=C:\Noa\StorageConnector\LocalTmp\

[Policy:3]
; this policy uses multiple paths:
; when the first path has reached the given size, the next path is used
Path1.Path=X:\Archive1\
Path1.Size=100MB

Path2.Path=X:\Archive2\
Path2.Size=100MB
```

The following variables are used:

- PolicyID: For each Media Format that should be mounted to the archive, the PolicyID needs to be linked to its Storage Properties (menu Catalog | Media Formats in mediARC GUI).
- Path#.Path: The exact path to the archive storage folder used for the given PolicyID. Both IP address and Hostname are allowed.
- Path#.Size: The maximum size of the given folder, expressed in integer decimal numbers and KB, MB, GB or TB units (no spaces/separators allowed). Use exact numbers, e.g. if the Path1.Size should be 18.7TB the correct value is "Path1.Size=19149GB" (18.7 * 1024).
- Path#.LocalTmpPath: The exact path to the StorageConnector's (local) temporary folder for the given PolicyID. The path will be used by StorageConnector e.g. for MD5 calculations of

files which are available on LTO tapes only and need to be retrieved first, or for temporarily storing files retrieved from a remote location via the RemoteFileAgent (RFA).

*Note: Depending on your network structure resolving hostnames might lead to application timeouts. If you encounter timeout problems upon application startup you should change the configuration to IP addresses.*

**StorageConnector DIVA:**

```
[Default]
DivaManager=127.0.0.1:7101
DivaLogin=user/password
MediaName=Default
SourceDestName=noa-gpp1
BasePath=\\Server01\GPP\

[Policy:1]
ObjectCategory=noa_hires
MediaName=DIVAGRID
Path1.SourceDestName=noa-gpp1
Path1.BasePath=\\Server01\GPP\
Path2.SourceDestName=noa-gpp2
Path2.BasePath=\\Server01\GPP\

[Policy:2]
ObjectCategory=noa_lowres
MediaName=DIVAGRID
Path1.SourceDestName=noa-gpp1
Path1.BasePath=\\Server01\GPP\
Path2.SourceDestName=noa-gpp2
Path2.BasePath=\\Server01\GPP\

[Policy:3]
ObjectCategory=noa_aux
MediaName=DIVAGRID
Path1.SourceDestName=noa-gpp1
Path1.BasePath=\\Server01\GPP\
Path2.SourceDestName=noa-gpp2
Path2.BasePath=\\Server01\GPP\
```

The following variables are used:

- DivaManager: IP address and port to connect to the Diva Manager.
- DivaLogin: Username and password to login to Diva Manager (leave empty if not required).
- MediaName: The tape group or disk array on which the object should be saved (a DIVA setting that should be provided by the DIVA administrator). A fallback value in case the entry is missing on Policy section level.
- SourceDestName: The destination for the restore from DIVA (usually GPP for further processing). A fallback value in case the entry is missing on Policy section level.

- PolicyID: For each Media Format that should be mounted to the archive, the PolicyID needs to be linked to its Storage Properties (menu Catalog | Media Formats in mediARC GUI).
- ObjectCategory: The category of the object (a DIVA setting that should be provided by the DIVA administrator). It's also used for archive path name in mediARC (see chapter mediARC File Access).
- MediaName: The tape group or disk array on which the object should be saved (a DIVA setting that should be provided by the DIVA administrator).
- Path#.SourceDestName: Destination for the restore from DIVA (usually GPP for further processing).
- Path#.BasePath: The exact path for the restore from DIVA (usually GPP for further processing).

### 13.3.2 Windows Registry

When the StorageConnector runs the first time, it generates the registry keys automatically. After stopping the StorageConnector service, the settings can be modified and get valid after the next service start.

The path to the StorageConnector settings stored in the Windows registry is:
- 32-bit Windows:
  HKEY_LOCAL_MACHINE\SOFTWARE\NOA\StorageConnector\
- 64-bit Windows:
  HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432node\NOA\StorageConnector\

The following registry keys are available:
- ActTransactionID: The ID number of the last transaction. This is a private entry and should not be modified!
- BindToIP: Specifies the IP network address to be used to listen to connection requests. If the hosting computer has several IP addresses, the StorageConnector can be enforced to bind it's TCP sockets to the specified address. Default value: '0.0.0.0'
- Compression: Specifies if the StorageConnector uses compressed client communication. Possible values: '0' and '1'. Default value: '0' (disabled).
- CustomMountFileCopyCmd: A customer-specific feature used by the UNC version of StorageConnector that allows to use a custom command-line executable for mounting the source file into the archive. The command string should include the path to the command-line executable plus the arguments. The placeholder string tags @@SourceFile@@ and @@DestFile@@ can be used within the argument part, the tags will be replaced by the (quoted) filenames when executing the command. Default value: none.
- DataRateRx: Obsolete, do not use! Default value: '0'.
- DebugLogFile: A valid path and file name for debug messages to be written to (e.g. "C:\STORCONlogs\STORCON.log"). If a file name is provided, debug messages will be written to this file. A new log file is started automatically by StorageConnector if the existing log file exceeds a size limit (currently 20MB). The amount and detail of debug messages depends on the setting of DebugLogLevel. If the file name is empty, no debug messages are written. Specifying a debug log file is recommended for debugging and problem analysis only. During normal operation, debug messages should be disabled. Writing a debug file may produce

significant amounts of data and may have an impact on application or system performance. It is recommended to watch storage resources when writing a debug log file. A local path is preferred over a server share to guarantee file access. Default value: none.
- DebugLogLevel: The level of detail for debug messages (on a scale from 1 to 5, where 5 represents the most detailed logging). A high level can result in a significant amount of data written. Debug levels should be used with care and debugging should be switched off when not needed. A general shortage in storage resources may influence system performance and stability. Default value: "3".

*Debug levels are:*
*"1" error messages*
*"2" warning messages*
*"3" normal messages*
*"4" info messages*
*"5" debug messages*

- Encryption: Obsolete, do not use! Default value: '0' (disabled).
- Encryption Key: Obsolete, do not use! Default value: none.
- InterfacePort: The IP interface port number for listening for client connections. Default value: "5707".
- LocalTmpPathMaxAge: Specifies a timeout in days for files stored on the local temp path. Note that this option is not available in all StorageConnector versions. Default value: '2'.
- LogExpirationDays: The max. age of a transaction log file in days, before it is deleted by the StorageConnector. Default value: '90'.
- LowTransactionPriority: File transactions are performed in threads. The threads have normal priority by default but can be set to a lower priority by setting the property to '1'. Note that this setting has only a limited effect on both file transfer speed and system resources usage. Default value: '1'.
- MaxConcurrentTransactions: The maximum amount of concurrently running file transactions to fine-tune the read/write performance of the storage. Default: '1'.
- PolicyFile: The complete path to the policy definition file that is used to map PolicyIDs to physical storage directories. Default value: 'C:\Noa\StorageConnector\PolicyFile\Policy.ini'.
- TransactionLogDir: The complete path to the transaction log files. If the directory does not exist, it is created on application startup. Default value: 'C:\Noa\StorageConnector\TransactionLogs\'.
- UseCustomCopyCmd: Enables or disables the use of CustomMountFileCopyCmd by the UNC version of StorageConnector. Default value: '0' (disabled).

### 13.3.3 mediARC File Access

The StorageConnector provides access to media files stored on the archive storage in the Archive Domain. In order for the archive storage to be available to the FileManager, the paths need to be defined in mediARC GUI using the menu Administration | File Access.

If StorageConnector UNC is used, the paths registered in File Access are identical to those defined in the StorageConnector's local 'Policy.ini' file. Example:

- Policy File:
  - o [Policy:1]
    Path1.Path=\\Server01\mediARC\Archive\BWF\
- File Access:
  - o Path: \\Server01\mediARC\Archive\BWF\

If StorageConnector DIVA is used, the paths names must match the ObjectCategory setting of the StorageConnector's local 'Policy.ini' file. Example:

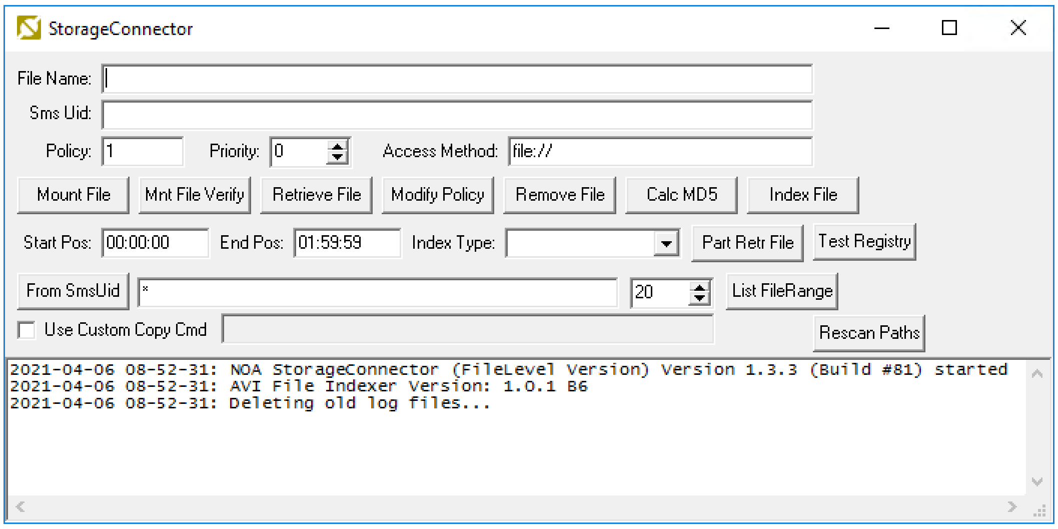
- Policy File:
  - o [Policy:1]
    ObjectCategory=noa_hires
    MediaName=DIVAGRID
    Path1.SourceDestName=noa-gpp1
    Path1.BasePath=\\server01\GPP\
    Path2.SourceDestName=noa-gpp2
    Path2.BasePath=\\server02\GPP\
    Path3.SourceDestName=noa-gpp3
    Path3.BasePath=\\server03\GPP\
- File Access:
  - o Path: noa-hires:\

*For further information about File Access configuration in mediARC, please refer to the mediARC Domains TechNote or the mediARC User Configuration and Rights Management Administrator Manual.*

## 13.4 StorageConnector GUI

*Warning: The StorageConnector GUI interface should only be used for debugging. If you are not sure what you are doing, stop and consult NOA support instead!*

While the StorageConnector is usually being run as a Windows service, it is possible to launch it as a GUI application.



```
2021-04-06 08-52-31: NOA StorageConnector (FileLevel Version) Version 1.3.3 (Build #81) started
2021-04-06 08-52-31: AVI File Indexer Version: 1.0.1 B6
2021-04-06 08-52-31: Deleting old log files...
```

*The StorageConnetor Window*

The following controls are available:

- File Name: The full path to the file that is to be processed by one of the available functions / buttons:
  - o in the case of Mount File it represents the full path to the source file that is to be mounted / copied to the archive storage (e.g. 'c:\test.avi').
  - o in the case of Retrieve File it represents the full path to the destination file that is to be created (e.g. 'c:\test.avi').
- Sms Uid: The full path:
  - o to which the file defined in File Name is to be copied by the Mount File function (e.g. '\\Server01\Archive\BWF\0F\0F2E\0F2E56\test.avi'). If no Sms Uid is supplied, StorageConnector will create a path based on the Policy path.
  - o of the file that is to be retrieved by the Retrieve Policy function.
  - o of the file that is to be modified by the Modify Policy function.
  - o of the file that is to be deleted by the Remove File function.
  - o of the file that is to be used for the MD5 calculation by the Calc MD5 function.
  - o of the file that is to be indexed by the Index File function.
- Policy: The policy to be used for the file operation. Default value: '1'.
- Priority: The priority of the file operation defined by an integer number between -2 and +2. Default value: '0'.
- Access Method: A customer-specific feature used by the UNC version of StorageConnector that allows RemoteFileAgent connections. When used, the source file is retrieved from RFA

unto the local temp path defined in the 'Policy.ini', and then moved to the archive. Possible values: 'file://' or 'RemoteFileAgent://<server>:<port>'

*Note that most of the below mentioned functions executed by the available buttons generate a log file which can be used for debugging.*

- Mount File: Transfer a file from local share into archive storage. The transfer is performed asynchronously by StorageConnector. When “SmsUid” is predefined (non-empty) it is used if possible. If not, the path and file name are generated. The optional 'Access Method' feature is not supported by all StorageConnector versions.
- Mount File Verify: Same as Mount File, except the file copy process is verified by reading back the mounted file from storage and comparing it to the given MD5 checksum.
- Retrieve File: Transfer a file from archive storage into a local share. The transfer is performed asynchronously by StorageConnector.
  - o If 'Start Pos', 'End Pos', 'Index Type' and 'Index File' are defined, the Retrieve File function will actually perform a Partial File Restore (PFR).
- Make Online: Transfer a file from offline into near-line (cache) storage within the archive system. The transfer is performed asynchronously by StorageConnector. This feature is not supported by all StorageConnector versions.
- Modify Policy: Change the file policy of an existing file within the archive system (and move the file to another generated path in the process). The processing is performed asynchronously by StorageConnector. This feature is not supported by all StorageConnector versions.
- Remove File: Delete a file within the archive system. The processing is performed asynchronously by StorageConnector.
- Calc MD5: Read the MD5 checksum of a file within the archive system. The processing is performed asynchronously by StorageConnector.
- Index File: Create an index file for partial file retrieve. Index data is written to “IndexDataFile”, if necessary. This feature is not supported by all StorageConnector versions.

- Start Pos: The timecode of the beginning of the selection for Partial File Retrieve (PFR), currently only supported for the StorageConnector DIVA.
- End Pos: The timecode of the end of the selection for Partial File Retrieve (PFR).
- Index Type: The type of index used for PFR of the file. Currently supported is only the NoaAviFileIndexer for DIVA media files. Normally, the index file is created by StorageConnector DIVA during Mount File and stored in the mediARC database, and is used on retrieval to generate a valid sub-range of the original file.
- Part Retr File: Opens a file browser window to select the file containing the index of the original file. The optional 'IndexFile' feature is not supported by all StorageConnector versions.

*Note that currently true Partial File Retrieve (PFR) is only available for DIVA. While the function can be tested with the StorageConnector UNC in GUI mode for debugging purposes, in production when the full file is available over CIFS, the MediaButler is used to do the sub-range read via ordinary file seek operations.*

- Test Registry: Obsolete, used to test whether user privileges allow writing a value to the Windows registry.

- From SmsUid: The path where the search for available files shall begin.
- Max File Count: The maximum number of results that shall be returned by the search.
- List FileRange: Search for and list all files available within the archive system. This feature is not supported by all StorageConnector versions.

- UseCustomCopyCmd: If checked, a custom command-line executable can be used for mounting the source file into the archive. The command string should include the path to the command-line executable plus the arguments. The placeholder string tags @@SourceFile@@ and @@DestFile@@ can be used within the argument part, the tags will be replaced by the (quoted) filenames when executing the command.

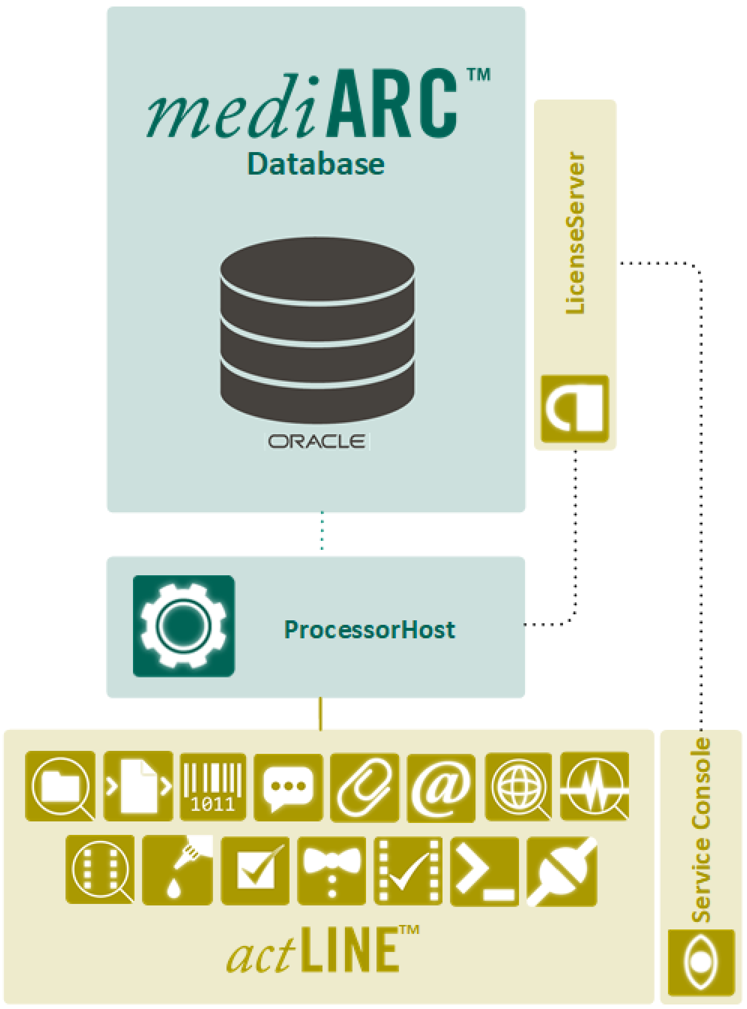
- Rescan Paths: Refresh the file index.

# NOA actLINE

# 14 NOA actLINE

actLINE gives users a powerful set of tools that allows them to reshape content during the archiving process while extending the effectiveness and power of NOA’s existing systems. Now, instead of using a combination of products from different vendors, the Customer can perform all of the functions in a workflow with one integrated system, which helps save time, ensure compatibility, and lower costs.

The actLINE software package contains tools for triggering workflows, moving files among workflows, decoding and transcoding files, recombining segmented archives, and much more. The tools are actually processors that perform specific functions within a workflow. By grouping together certain processors in certain combinations, actLINE helps the system process tasks more efficiently. actLINE applications connect to the mediARC system through the ProcessorHost.



The following modules are part of the actLINE product family:

| | |
|---|---|
| AutoCut | automatically aligns and concatenates split or segmented recordings coming from NOARecord into a single, digital file. |

| | |
|---|---|
| BarcodeStation | allows to execute actions triggered by a barcode scan. |
| CLIP | a generic processor that can interface with any 3rd party product that allows control via a command line prompt, to execute operating system commands within a mediARC workflow. (Command Line Interface Processor\| |
| DBScripter | a script interpreter for executing scripts in mediARC. |
| EMailer | a processor application built into a workflow that notifies users of specific workflow conditions. |
| FileCollector | moves the files detected by the FolderScanner to a public processing path. |
| FolderScanner | a monitoring application that continuously scans pre-determined folders on an import client for new media and metadata files. |
| LicenseServer | allows to count and control the amount of granted licenses. |
| MediaButler | a generic media transcoding processor that can perform multiple audio and video transcodings. |
| MetaDataFinder | checks different pre-configured web data sources for metadata. |
| QualityChecker | allows manual quality control which allows to watch, check, edit and rate the ingested media content. |
| ServiceConsole | provides a centralized, real-time overview of nearly all NOA processors running on distributed servers. |
| StorageConnector | a file access agent, which handles the access of archive storage to the rest of the system. |
| UniversalDialoger | allows to present and collect data as a workflow task in a dialog. |
| VideoFileAnalyzer | an interface to common 3rd party analyzer tools for video container/codec quality analysis. |
| VideoScanner | a video file normalizing gateway, translating a typically lossy production format to a unified, mathematically lossless, and open |

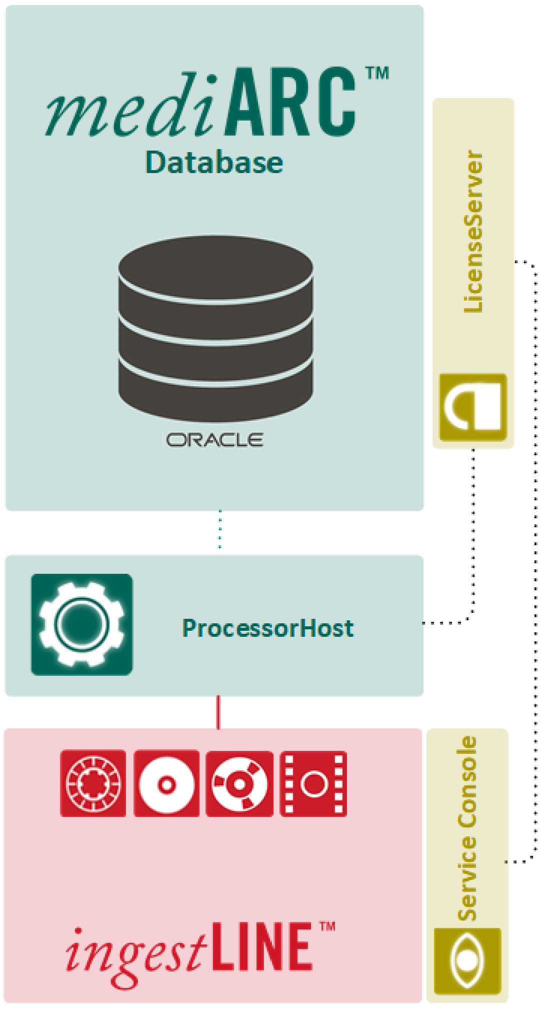
| | |
|---|---|
| | mezzanine format. |
| WaveScanner | an audio normalizing gateway, which analyzes and scans a variety of audio files for quality, decodes them optionally towards a linear wave file, and reads out metadata. |

*For further information, please refer to the actLINE documentation.*

# NOA ingestLINE

# 15 NOA ingestLINE

The ingestLINE media transcription system from NOA allows users to digitize extensive media archive collections without introducing new errors from faulty transfer parameters. ingestLINE software components notify users of critical transfer problems during transcription. This quality-related transfer information – collected at the moment of ingest – is then stored within ingest reports of mediARC for further workflow management.



The following modules are part of the ingestLINE product family:

| | |
|---|---|
| NOARecord | a multi-stream ingest and transcription software for analog audio legacy sources with full QC control. |
| MediaLector | an ingest tool for quality-controlled mass migration of DATs and MiniDiscs from one to eight parallel stations. |
| CDLector | a mass CD-ripping system that is capable of handling up to eight parallel drives. |

| | |
|---|---|
| FrameLector | an ingest tool for efficient and quality controlled mass transcription of video tapes. |

*For further information, please refer to the ingestLINE documentation.*

# Glossary

# 16 Glossary

| | |
|---|---|
| Access Domains | A mediARC domain where all members have the same physical and logical possibilities for accessing a file. |
| Access rights | The permissions an individual user or a computer application holds to read, write, modify, delete or otherwise access a computer file. |
| Active Directory | A directory service developed by Microsoft for Windows domain networks. It authenticates and authorizes all users and computers in a Windows domain type network. |
| Annotation | The practice and the result of adding descriptive metadata to objects in mediARC. |
| API | An interface that defines interactions between mediARC and multiple custom 3rd party software applications. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. |
| Archive Asset Management (AAM) | A category of software that takes care of centralizing, standardizing and distributing digital archive files. Usually an AAM system manages storing, ingesting, organizing, indexing, searching, processing, outgesting and retrieving, etc. of digital archive assets (both files and metadata), incl. version control and access control. |
| Archive Assets | Archive assets represent relevant data that exists in a digital format, has great value, and generally comes with the right to use. In mediARC archive assets are objects that consist of media, metadata and rights. |
| Archive Catalogue | A systematic (usually relational) list of metadata items in an archive. |
| Archive Domain | The mediARC Access Domain that contains the archive, resident inside an HSM or a disk storage. |
| Authentication | The act of proving the identity of a computer system user. |
| Block level storage | A concept in cloud-hosted data persistence where cloud services emulate the behavior of a traditional block device, such as a physical hard drive. Storage in such is organized as blocks. This emulates the type of behavior seen in traditional disks or tape storage through storage virtualization. |
| Cache | A hardware or software component that stores data so that future requests for that data can be served faster. |
| Database | An organized collection of data, stored and accessed electronically from a computer system. |
| Database dump | A record of the table structure and/or the data from a database which is most often used for backing up a database so that its contents can be restored in the event of data loss. |
| Database patch | A set of changes to a database designed to update, fix, or improve it. |
| Database schema | The structure of a database, i.e. the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases like the one used by mediARC). |
| Dongle | A hardware device used for digital rights management, which typically contains a license key or some other cryptographic protection |

| | |
|---|---|
| | mechanism. |
| Explicit users | mediARC users who are not members of the Windows domain but are nevertheless granted access by the LicenseServer based on its local user database. |
| Failover connection | A separately defined connection to the mediARC database used for connecting to the same database via another network route. |
| GarbageCollector | An automatic cleanup process managed by the ProcessorHost which removes files created by processor applications that match a certain pre-defined set of conditions (e.g. age). |
| General Public Path (GPP) | mediARC's central high-performance system storage, predominantly used by Processors for task and workflow processing. |
| GUI application | A software program with a graphical user interface that allows users to interact with it through graphical controls (as opposed to text-based CLI interfaces or Windows services running without an interface on the background). |
| Checksum | A small-sized block of data used to verify data integrity. |
| Ingest | The process of importing data or other material into a system. |
| Instance | A separately running copy of the same application. |
| JIRA | The issue tracking system used by NOA. |
| LDAP | An open and cross platform protocol used for directory services authentication. (Lightweight Directory Access Protocol) |
| License | An official permit to use a specific piece of software. |
| License model | Licensing approaches officially supported by NOA. Currently "restrictive" and "warning". |
| License peak | The number of exceeded licenses. |
| Markers | Metadata records describing time based events associated with and linked to Media Items in mediARC, that are mostly generated automatically based on Event Type templates (e.g. cue marker). |
| Media | Content that is stored in the system's media archive as essence data. |
| Media essence | The coded video and audio inside a video or audio file container (i.e. not headers, footers, and metadata). |
| Metadata | Descriptive data of both general and media-specific nature. |
| NoaComm | A proprietary NOA communications protocol for high-level remote procedure calls between programs in a mediARC system. |
| OAIS | The ISO OAIS Reference Model represents the optimum standard to create and maintain a digital archive repository over a long period of time. |
| Outgest | The process of exporting data or other material from a system. |
| Prelisten | An audible reproduction of audio content stored in the archive. |
| Prelisten Cache | A shared network directory in a mediARC User Domain where the prelisten/preview media files used for streaming reside. |
| Preview | An on-screen view of video content stored in the archive. |
| Processor | A program or a program part, which can alter the state of a Task. It runs on a Computer, which is the physical workstation. |
| Round robin | An algorithm for assigning equal access to different resources in turns. |

| | |
|---|---|
| SID | A unique, immutable identifier of a user, user group, or other security principal in a Microsoft Windows operating system. (Security Identifier) |
| PL/SQL | Oracle Corporation's procedural extension for SQL, a domain-specific language used for managing data held in a relational database management system. (Procedural Language for SQL) |
| System Domain | The mediARC Access Domain that contains technology responsible for media and metadata processing, ingesting, and outgesting. |
| Task | A unit of execution in a mediARC workflow. |
| Throughput | The sum of the data rates that can be delivered to all clients of a system. |
| Timeout | A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. |
| TNS | A proprietary Oracle computer-networking technology, that operates mainly for connection to Oracle databases. (Transparent Network Substrate) |
| Transcoding engine | Software used for the direct digital-to-digital conversion of one encoding to another, typically in cases where a target device (or workflow) does not support the format or has limited storage capacity that mandates a reduced file size, or to convert incompatible or obsolete data to a better-supported or modern file format. |
| Unicode | An information technology standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems, maintained by the Unicode Consortium. |
| URL | Colloquially termed a "web address", URL is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. |
| User Domain | The mediARC Access Domain that contains the users of the archive who are allowed to see and hear content via proxies and get access to a copy of archive files. |
| User Rights Management (URM) | A security feature for controlling user access to specific parts or features of a mediARC system that would normally be restricted to the administrator role. |
| User synchronization | A process that copies user and group information from an organization's enterprise directory system (such as an Active Directory) to mediARC's internal user database. |
| User token | Contains the security credentials that identify the user for a login session. |
| UTF-8 | a variable-width character encoding used for electronic communication (Unicode Transformation Format – 8-bit) |
| WIBUBOX | A dongle based software which controls licensing for mediARC's LicenseServer. |
| Workflow engine | Configurable software that manages specific tasks and processes inside mediARC. |
| Workflow | Pre-configured processes managing any transaction in the system like ingest or retrieval of archive content. |

# Impressum

# 17 Impressum

## Copyright

Printed: May 2021 in Vienna, Austria.

## Disclaimer

## Version Info

This document targets mediARC version 1.7.3 build 404.

## Publishing Info

This document was published on 18 May 2021.