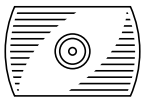
HighFinesse

The Standard of Accuracy

User Guide

HighFinesse NetAccess, Network Solution (beta)



HighFinesse
Wavelength Meter (WS-Series)

HighFinesse
Laser Spectrum Analyzer (LSA)

HighFinesse GmbH
Laser and Electronic Systems

Wöhrdstraße 4
72072 Tübingen, Germany

Phone: +49 (0) 70 71 53 918 0
Fax: +49 (0) 70 71 53 918 99
Email: info@highfinesse.com

www.highfinesse.com

Executive Director:
Dr. Thomas Fischer

Commercial Registry Stuttgart, Germany, **HRB** 382133

Contents

Contents

Document Date: 2022-11-15

Scope of document

HighFinesse_NetAccess_YYYYMMDD.zip contains following beta software components:

- wlmDataServer.exe versions: ≥ 7.32/64.263.4
- wlmData.dll version: wlmData.dll version: ≥ 7.263.4
- wlmData-M.m.b-ARCH.deb, .rpm, .sh and .tar.gz packages version: ≥ 7.263.4
- wlmData-M.m.b-ARCH.pkg macOS installer package ≥ 7.263.4

1. Introduction

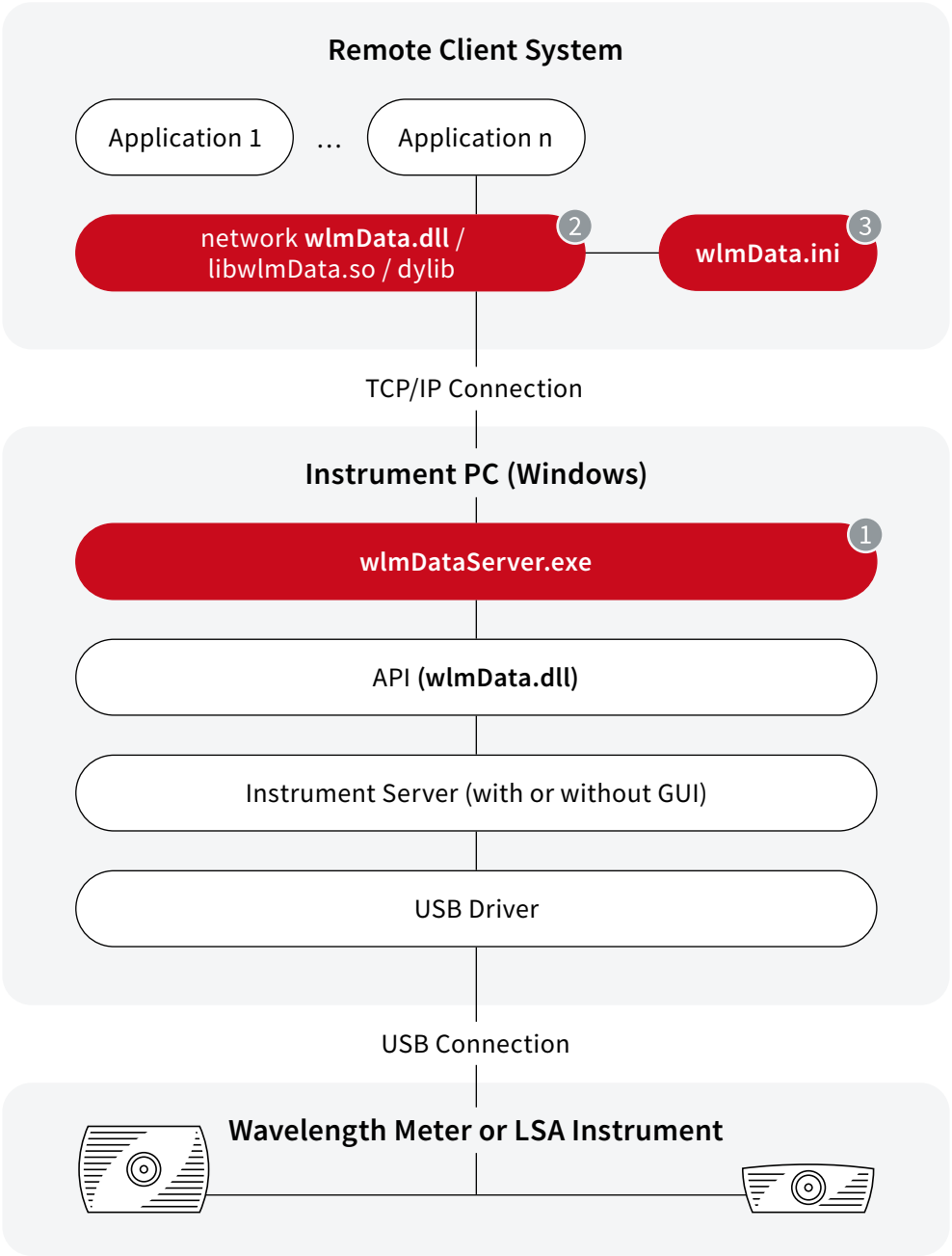
NetAccess is a fast and lightweight networking extension for the existing Wavelength Meter (WLM) and Laser Spectrum Analyzer (LSA) product lines. Contrary to the conventional remote desktop (VNC, Teamviewer, etc.) applications, it provides compatible API's on the remote side over reliable TCP/IP networks; therefore the existing / legacy softwares can be run without any modification or selfmade network solutions.

Typical applications:

- Test and measurement automation
- Remote laboratories
- Measurement instrument sharing

1.1. Components

NetAccess network solution is based on the Client-Server model and contains the following software components (highlighted in red):



1. **wlmDataServer.exe** executable (server side)
2. **wlmData.dll, libwlmData.so** or **libwlmData.dylib** network API library (client side)
3. wlmData.ini (configuration file for the wlmData network API)

Figure 1: NetAccess software components

2. Installation

2.1. Compatibility

The software components are tested on the following Operating systems and (architectures):

- **wlmDataServer.exe** and **wlmData.dll:** Windows 10 (Intel 32/64 bit)
- **libwlmData.so:** Ubuntu 18.04 LTS / 20.04 LTS, Fedora 29 and Debian 9 (Intel 64 bit, ARM 32/64 bit: armv7l, aarch64)
- **libwlmData.dylib:** macOS Big Sur version 11.2.1 (Intel 64 bit, M1)

The Linux ARM 32/64bit support makes possible to use single-board computers like RedPitaya, RaspberryPi or Nvidia Jetson family on client side.

2.2. Network Recommendation

Virtual private network (VPN) or a separate Fast or Gigabit Ethernet measurement network segment is recommended by reason of security and performance.

2.3. Package Contents

The installation package structured in the manner shown in Figure 2 (next page).

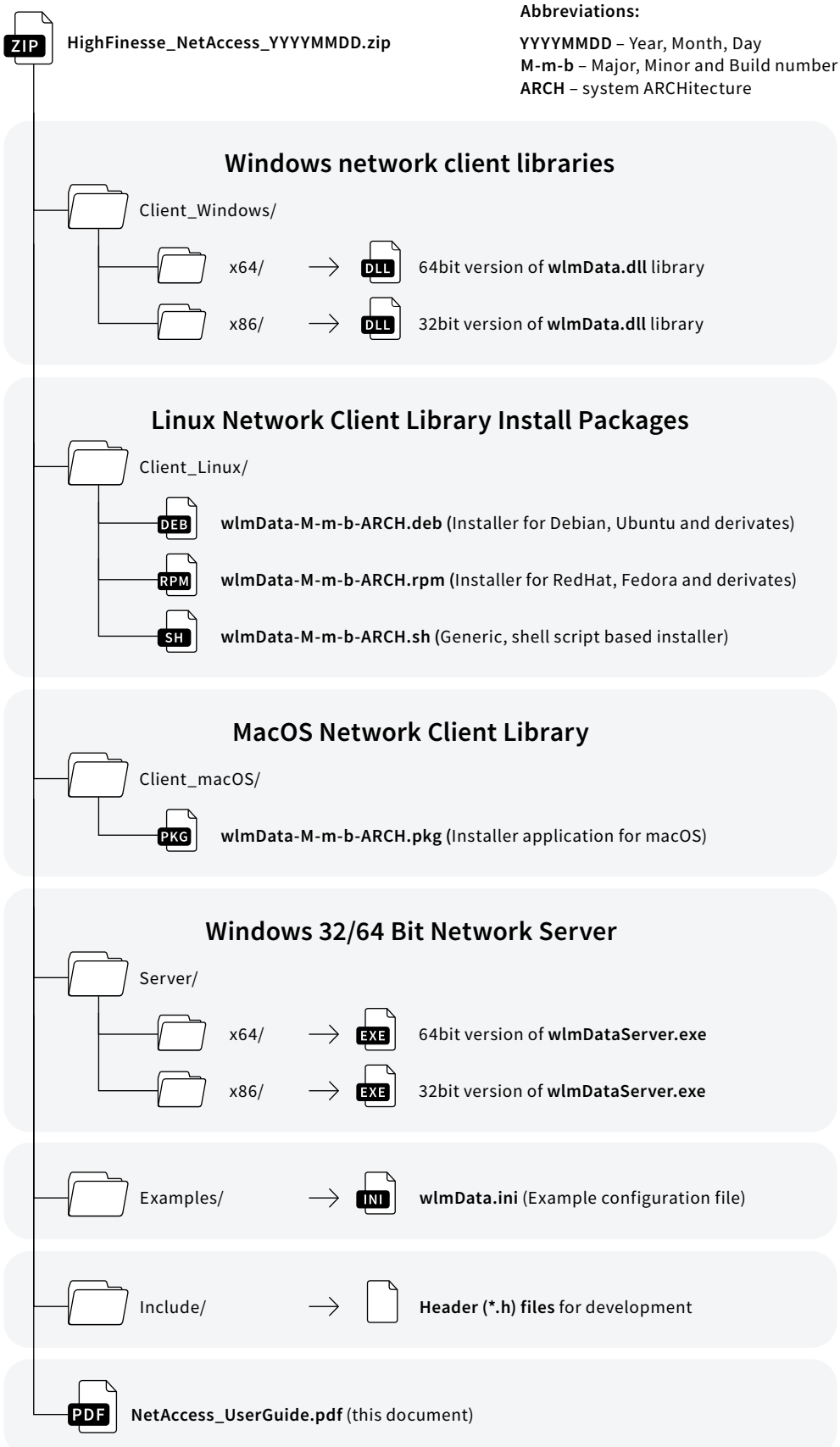


Figure 2: Package contents

2.4. Server Side Installation

2.4.1. Prerequisites

1. The proper 32/64 bit version of Visual C++ Redistributable for Visual Studio 2017 is required to run **wlmDataServer.exe:**
- **vc_redist.x86.exe (32 bit)**
- **vc_redist.x64.exe (64 bit)**

https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

2. An installed Wavelength Meter (WLM) or Laser Spectrum Analyzer (LSA) software package

2.4.2. Server Side Installation

The server side implemented as a portable application, and it can be run from any location. Copy the **wlmDataServer.exe** into a directory of choice on the Instrument PC. At the first start, allow the firewall access for the **wlmDataServer.exe** at the Windows Security Alert dialog.

2.5. Client Side Installation – Windows

2.5.1. Prerequisites

1. The proper version of Visual C++ Redistributable for Visual Studio 2017 is required to use networked version of **wlmData.dll:**
- **vc_redist.x86.exe (32 bit)**
- **vc_redist.x64.exe (64 bit)**

2.5.2. System Wide Installation

In case of using 32 bit Windows operating system:

Copy the 32 bit version of **wlmData.dll** into the folder:
%SYSTEMROOT%\System32

In case of using 64 bit Windows operating system:

Copy the 32 bit version of **wlmData.dll** into the folder:
%SYSTEMROOT%\SysWoW64

Copy the 64 bit version of **wlmData.dll** into the folder:
%SYSTEMROOT%\System32

⚠ **PLEASE NOTE:**

32/64-bit versions of wlmData.dll are interoperable with
32/64 bit wlmDataServer.exe

2.6. Client Side Installation – Linux

2.6.1. System Wide Installation

The system wide deployment requires root privileges / password.
Copy the **appropriate deb** / **rpm** / **sh package** into the home directory and use the following command from terminal prompt:

- **Debian, Ubuntu or derivative distributions:**

```
[user@host]$ su
Password:
[root@host]# dpkg -i ./wlmData-M-m-b-ARCH.deb
[root@host]# exit
```

- **RedHat, Fedora or derivative distributions:**

```
[user@host]$ su
Password:
[root@host]# rpm -i ./wlmData-M-m-b-ARCH.deb
[root@host]# exit
```

- **In generic case:**

```
[user@host]$ su
Password:
[root@host]# chmod +x ./wlmData-M-m-b-ARCH.sh
[root@host]# ./wlmData-M-m-b-ARCH.sh -prefix=/usr -exclude-subdir
[root@host]# exit
```

2.6.2. User level Installation

Without root permissions the client library can be deployed to the user's home directory.

Copy the appropriate **deb** / **rpm** / **sh package** into the home directory and use the following command from terminal prompt:

- **Debian, Ubuntu or derivative distributions:**

```
dpkg -x ./wlmData-M-m-b-ARCH.deb ./
```

- **RedHat, Fedora or derivative distributions:**

```
rpm2cpio ./wlmData-M-m-b-ARCH.deb | cpio -i -d
```

- **In other case:**

```
chmod +x ./wlmData-M-m-b-ARCH.sh
./wlmData-M-m-b-ARCH.sh -prefix=./usr -exclude-subdir
```

⚠ **PLEASE NOTE:**

32/64 bit of **wlmData.so** is interoperable with
32/64 bit **wlmDataServer.exe**

2.7. Client side installation – macOS

- The installation requires Admin user / password.
- Copy the **wlmData-M-m-b-ARCH.pkg** installer package to the computer.
- Double click on package icon and follow the instructions of GUI based installer.

3. Operation

3.1. Server (Instrument) Side

The server side of the solution is the **wlmDataServer.exe** program. Starting without any additional parameter it will listen on TCP port 7171 and 7172, IPv4/6 addresses of using all available network interfaces including the loopback adapter (127.0.0.1/::1).

The default behaviour can be modified by the following command line parameters and switches:

```
/p<Set/Get function TCP port>
/c<CallBack function TCP port>
/a<IPv4/6 address>
/v<verbosity level [0-6]>
/? Usage
```

Verbosity levels are:

Cfg.	Level	Description
0	NONE	No messages
1	FATAL	Any error that makes impossible the application run
2	ERROR	Any error that prevents the network service
3	WARNING	Potential problems
4	INFO	Useful information about normal operation (Default setting)
5	DEBUG	Displays API function calls
6	TRACE	Most detailed, protocol level information for debugging

Table 1: Server side verbosity levels

Find an example on the next page (Figure 3).

EXAMPLE:

wlmDataServer.exe to be listening on 192.168.88.10 IPv4 address using 3132, 3133 TCP ports, verbosity level: debug.

```
wlmDataServer.exe /p3132 /c3133 /a192.168.88.10 /v5
```

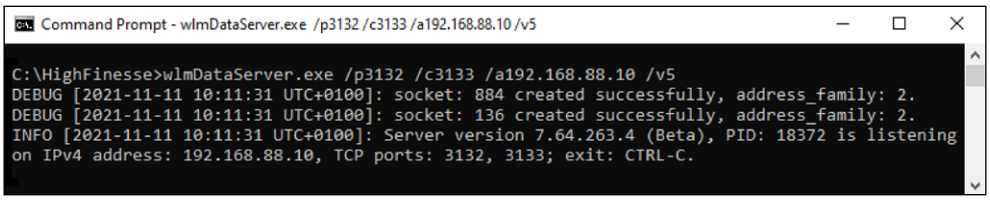


Figure 3: Starting wlmDataServer.exe with parameters

⚠ PLEASE NOTE:

Unique port numbers must be used for each server instances.

3.2. Network Client Library

3.2.1. Compatibility

The network version of the wlmData.dll (Windows) libwlmData.so (Linux) or libwlmData.dylib (macOS) is designed to be drop-in compatible with the original API, and can be re-used by existing code or applications without modification.

Compatible features:

- Full support of Set/Get functions.
- Full support of user defined CallbackProc/CallbackProcEx procedures.
- Full support of raw data transfer GetPattern(), GetAnalysis().. functions.

⚠ PLEASE NOTE:

For the detailed API information please refer to the instrument's User Manual.

3.2.2. Error Signaling Modes

To ensure the highest level of compatibility with existing applications, there is no network error specific return code in the API. Network error detection can be done in the following ways:

Cfg.	Mode	Description
0	NONE	Errors are discarded
1	LOG	Errors are sent to the Logging system (see bellow)
2	DIALOG	Errors are be displayed as a Dialog Box (Windows only)
4	CALLBACK	Errors are signaled via user provided callback function
8	EXIT	Application will be terminated with error specific exit code

Table 2: Error signaling mode configuration codes (bit fields)

⚠ **PLEASE NOTE:**

Configuration codes are bit fields, more signaling modes can be used in parallel.

Default configuration:
The default setting for error signaling is LOG + EXIT (1+8=9) which can be modified in **wlmData.ini.**

3.2.3. Error Identification

The network client library uses triplets for identifying **Cause, Source** and **Error code** of the problem.

Cause IDs:

ID	Mode	Description
2	SEND_NETOWRK	Socket function send() returned with error code
3	SEND_SIZE	Socket function send() transferred wrong number of bytes
4	RECV_NETWORK	Socket function recv() returned with error code
5	RECV_SIZE	Socket function recv() transferred wrong number of bytes
6	SESSION_ID	Session ID check failed
7	MAGIC_NUMBER	Protocol specific magic number check failed
8, 9		Reserved for future usage
10	CONN_NETWORK	Socket function connect() returned with error code

Table 3: Error cause IDs

Source IDs:
Source IDs are unique to API functions. For the full list of Source IDs please refer to the **wlmDataNetworkConstants.h** header file.

Error codes:

Cause	Error code
SEND_NETOWRK, RECV_NETWORK, CONN_NETWORK	Socket (OS) specific return code of connect(), send() or recv() function
SEND_SIZE, RECV_SIZE	Number of transferred bytes
SESSION_ID	Client side session ID
MAGIC_NUMBER	Client side magic number

Table 4: Error codes

3.2.4. Log Levels

The logging feature uses verbosity levels as the server side:

Cfg.	Level	Description
0	NONE	No messages
1	FATAL	Any error that makes impossible the application run
2	ERROR	Any error that prevents the network service
3	WARNING	Potential problems
4	INFO	Useful information about normal operation
5	DEBUG	Detailed information for debugging
6	TRACE	Most detailed, protocol level information for debugging

Table 5: Client side verbosity levels

Default configuration:
The default loglevel is WARNING (3), which can be modified in **wlmData.ini.**

3.2.5. Log Modes

The network client library has logging feature which uses any combination of the followings outputs:

Cfg.	Mode	Description
0	NONE	Messages are discarded
1	STDERR	Messages will be sent to the Standard Error Output
2	STDOUT	Messages will be sent to the Standard Output
4	DIALOG	Messages will be displayed as a Dialog Box (Windows only)
8	CALLBACK	Messages are transferred via user provided callback function

Table 6: Log mode configuration codes (bit fields)

Default configuration:
The default logging output is STDERR (1).
The default setting can be modified in **wlmData.ini.**

3.2.6. User Provided Callback Functions

The log and error behavior of the network API library can be customized by user-specified callback functions. Attaching and detaching is done by the appropriate Mode parameter of the Instantiate() API function.

```
long Instantiate(long RFC, long Mode, long *P1, long P2);
```

Parameters for attaching / detaching callbacks:

```
RFC:    Use cInstNotification constant

Mode:   cNotifyInstallLogEvent    = 1000
        cNotifyInstallErrorEvent  = 1001
        cNotifyRemoveLogEvent     = 1002
        cNotifyRemoveErrorEvent   = 1003

P1:     Pointer to the callback function

P2:     Don't Care, can be set to zero
```

Return value:
Function returns 1 in the case of success, otherwise 0.

3.2.6.1. Log Callback Function

Declaration:

```
void vLogCallBack(int32_t xVerbosityLevel, const char* pcLogMessage);
```

Parameters:

`xVerbosityLevel`:	Numerical code of Log-event severity (see section: 3.2.4.)
`pcLogMessage`:	Pointer to the log message string terminated with zero, with maximum length of 1024 bytes

Return value:
vLogCallBack() function has no return value.

3.2.6.2. Error Callback Function

Declaration:

```
void vErrorCallBack(int32_t xCause, int32_t xSource, int32_t xCode);
```

Parameters:

`xCause`: Numerical code of error cause (see section: 3.2.3.)

`xSource`: Unique ID of error source (see section: 3.2.3.)

`xCode`: Cause and OS dependent error code (see section: 3.2.3)

Return value:
vErrorCallBack() function has no return value.

3.2.7. Network connection control

By default, the network API library automatically initiates, maintains and closes the network connection to the server (autoconnect = 1), which works well for the vast majority of applications. In other cases, the connection state and parameters can be controlled manually (autoconnect = 0) using the Instantiate() function.

```
long Instantiate(long RFC, long Mode, long *P1, long P2);
```

Parameters for network connection control:

```
RFC:       Use cInstNetworkControl = 6 constant

Mode:      cSetConnectState = 1000
           cGetConnectState = 1001
           cSetParameter    = 1002
           cGetParameter    = 1003

P1:        Pointer to the parameter, direction and type
           depends on Mode parameter. Please refer to the
           wlmDataNetworkConstants.h header file for the
           details.

P2:        Don't Care, can be set to zero
```

Return value:
Function returns 1 in the case of success, otherwise 0.

⚠ PLEASE NOTE:

The modified connection parameters will be applied at next connection attempt.

3.3. wlmData.ini Configuration File

wlmData.ini is optional and contains application level configuration for the networked API library.

3.3.1. Search Path and Order – Windows

Search path order for the wlmData.ini file on Windows operating systems:

1. Application startup directory
2. DLL's directory
3. %LOCALAPPDATA%\HighFinesse\wlmData**wlmData.ini**

3.3.2. Search Path and Order – Linux and macOS

Search path order for the wlmData.ini file on Linux operating systems:

1. Application startup directory
2. $HOME/.config/HighFinesse/**wlmData.ini**
3. /etc/HighFinesse/**wlmData.ini**

3.3.3. File Format

wlmData.ini is a human readable configuration file that follows the widely used informal ini file standard with **properties** and **sections.**

Comments:
Comment lines are ignored and can be marked with semicolon (;) at the beginning of the line.

```
; Comment line will be ignored
```

General property syntax:
Every property has a name and value filed. Fields are case sensitive and delimited by equals sign (=)

```
; Property field with name and value
name = value
```

General section syntax:
Property fileds can be grouped into sections, which are to be defined with square brackets:

```
; Section1 contains name1 and name2 property fields.
[Section1]
name1 = valueA
name2 = valueB

; Section2 contains name1 and name2 property fields with
different values.
[Section2]
name1 = valueC
name2 = valueD
```

3.4. Settings

The following predefined properties and section names can be used in the **wlmData.ini:**

Properties:

Name	Values	Default	Description
version	[4|6|46|64]	4	prefered IP version if hostname is given
address	IPv4|IPv6|name	127.0.0.1	IPv4/6 address or hostname of the wlmData server
port	[0 – 65535]	7171	Set/Get function TCP port number
port2	[0 – 65535]	7272	Callback function port number
offload	[0 | 1]	0	Client side offload of Convert..() functions [0-off | 1-on]
errormode	[0 – 15]	9	Selected error mode(s) (see section: 3.2.2.)
logmode	[0 – 15]	5	Selected log mode(s) (see section: 3.2.5.)
loglevel	[0 – 6]	3	Selected loglevel (see section: 3.2.4.)
autoconnect	[0|1]	1	Server connection method (see section 3.2.7)

Table 7: wlmData.ini defined property names

Sections:

wlmData.ini uses sections for application name based configuration.
The “default” section fits all programs, otherwise use the application name for the specific settings.

```
; default settings for all application
[default]
address = 192.168.88.10

; Settings for Longterm.exe application
[Longterm.exe]
address = 192.168.88.12
loglevel = 6
```

Default configuration:

By default the networked client library tries to connect to a local server over IP 127.0.0.1, using TCP ports: 7171 and 7172.

4. Configuration Examples

4.1. Scenario 1 – Instrument Sharing

The Wavelength Meter is shared on the measurement network (192.168.10.0/24) by the Instrument PC. The WLM can be used by more clients in time multiplex way, the second client can be connected after the disconnection of the first client.

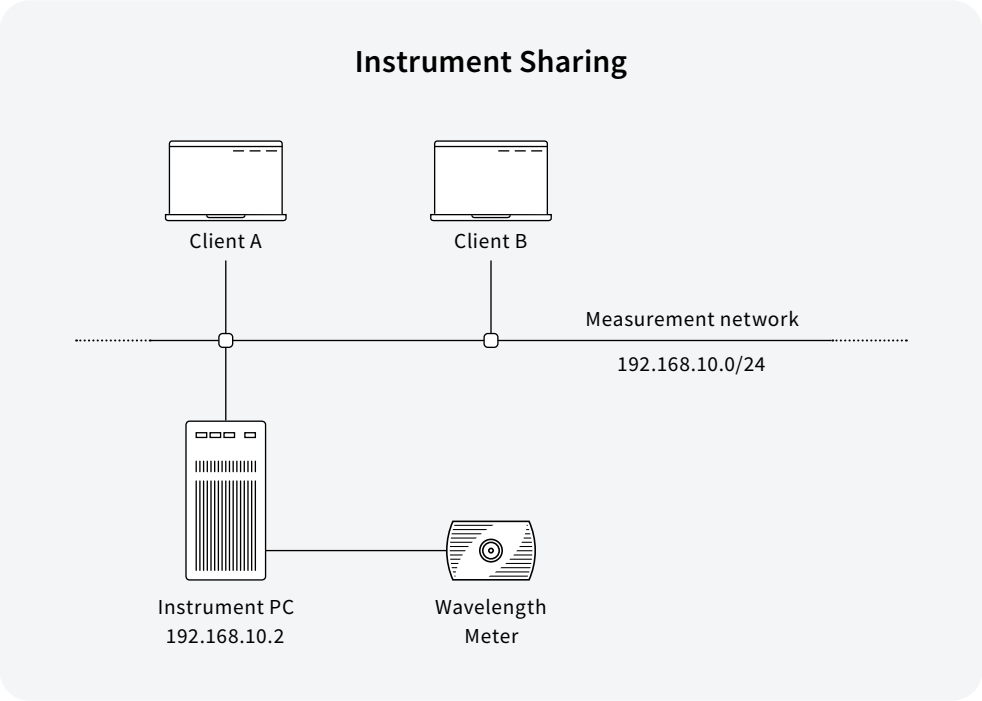


Figure 4: Scenario 1 – instrument sharing

Instrument PC configuration:
Start the **wlmDataServer.exe** on the instrument server without any additional command line switches.

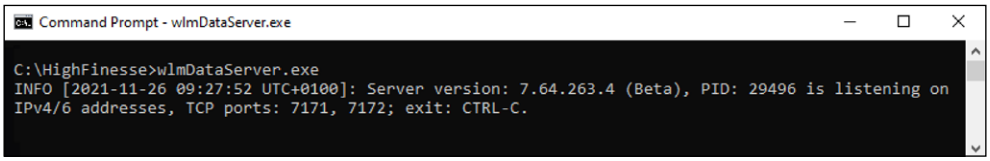


Figure 5: Starting wlmDataServer.exe with no parameters

Client configuration:
Use the following wlmData.ini file on Client “A” and Client “B”

```
; ============================== CUT HERE ==============================

; wlmData.ini example scenario 1 configuration file

[default]               ; Default settings (it match with
                          all measurement applications)
address = 192.168.10.2  ; Instrument server IP address
port    = 7171          ; Set/Get TCP Port number
port2   = 7172          ; CallbackProc/Ex TCP Port number
offload = 1             ; ConvertUnit / ConvertDeltaUnit functions
                        ; network offload (1=On, 0=Off)

; ============================== CUT HERE ==============================
```

4.2. Scenario 2 – Multiple Instrument Access From Single Client

Two Wavelength Meters (1 + 2) are shared on the measurement network (192.168.10.0/24) by Instrument PC 1 (192.168.10.2) and Instrument PC 2 (192.168.10.3).The client runs **“LongTerm1.exe”** and **“LongTerm2.exe”** for controlling Wavelength Meter 1 (1) and Wavelength Meter 2 (2).

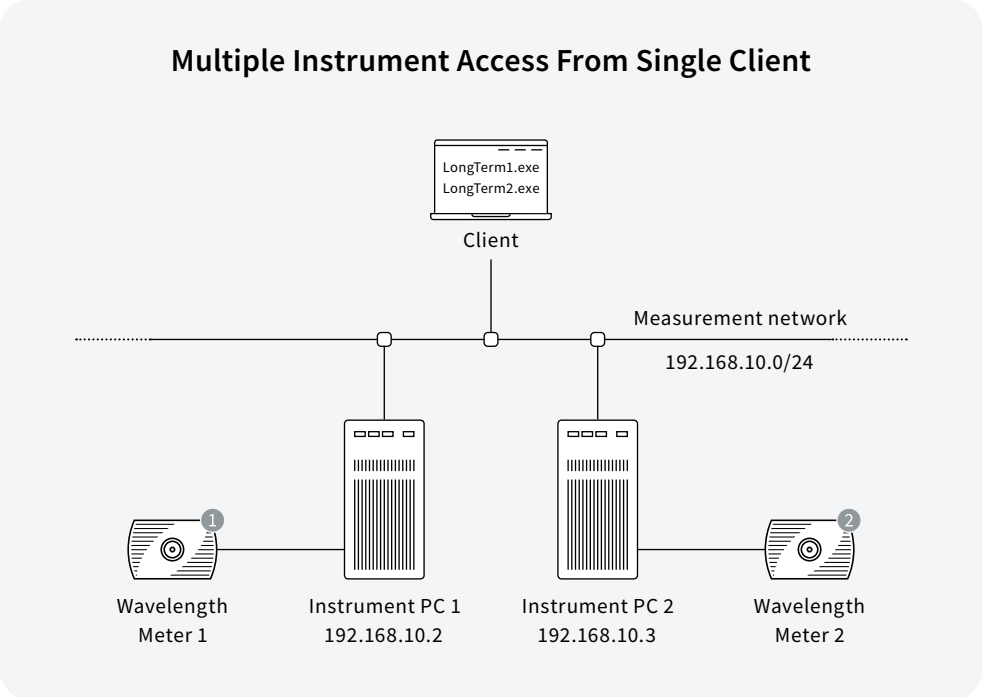


Figure 6: Scenario 2 – Multiple Instrument access from one client

Instrument PC configuration:
Start the **wlmDataServer.exe** on the both Instrument PCs without any additional command line switches.

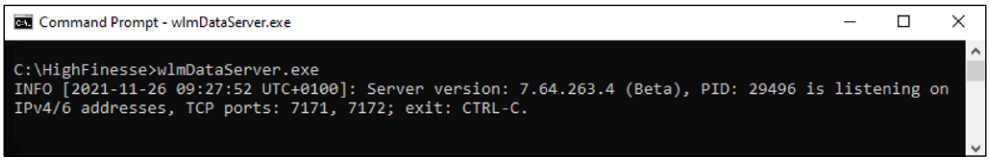


Figure 7: Starting wlmDataServer.exe with no parameters

Client configuration:
Use the following **wlmData.ini** file on Client side:

```
; ============================== CUT HERE ==============================

; wlmData.ini example scenario 2 configuration file

[LongTerm1.exe]          ; Configuration section for LongTerm1.exe
address = 192.168.10.2   ; Instrument server IP address
port    = 7171           ; Set/Get TCP Port number
port2   = 7172           ; CallbackProc/Ex TCP Port number
offload = 1              ; ConvertUnit / ConvertDeltaUnit functions
                           network offload (1=On, 0=Off)
;

[LongTerm2.exe]          ; Configuration section for LongTerm2.exe
address = 192.168.10.3   ; Instrument server IP address
port    = 7171           ; Set/Get TCP Port number
port2   = 7172           ; CallbackProc/Ex TCP Port number
offload = 1              ; ConvertUnit / ConvertDeltaUnit functions
                           network offload (1=On, 0=Off)

; ============================== CUT HERE ==============================
```

5. Open Source Licences

The network client uses “inih” library:
https://github.com/benhoyt/inih/

The “inih” library is distributed under the New BSD license:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Ben Hoyt nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

6. Legal Disclaimer – Product Liability

HighFinesse GmbH has no liabilities (and, to the HighFinesse's knowledge, there is no basis for any present or future action against HighFinesse GmbH giving rise to any liability) arising out of any injury to individuals or property as a result of ownership, possession or use of any product designed, manufactured, sold, leased or delivered by or any services performed or delivered by HighFinesse GmbH.

HighFinesse

The Standard of Accuracy

HighFinesse GmbH
Laser and Electronic Systems

Wöhrdstraße 4
72072 Tübingen, Germany

For further information please
feel free to contact us.

Phone: +49 (0) 70 71 53 918 0
Fax: +49 (0) 70 71 53 918 99
Email: info@highfinesse.com

www.highfinesse.com