Quick Start Guide

MVI69E-LDM-TWx
ThingWorx

❖ ProSoft Technology®

March 16, 2020

Important Information

Your Feedback Please

We always want you to feel that you made the right decision to use our products. If you have suggestions, comments, compliments or complaints about our products, documentation, or support, please write or call us.

ProSoft Technology, Inc.
+1 (661) 716-5100
+1 (661) 716-5101 (Fax)
www.prosoft-technology.com
support@prosoft-technology.com

March 16, 2020

ProSoft Technology®, is a registered copyright of ProSoft Technology, Inc. All other brand or product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

In an effort to conserve paper, ProSoft Technology no longer includes printed manuals with our product shipments. User Manuals, Datasheets, Sample Ladder Files, and Configuration Files are provided at our website: www.prosoft-technology.com

Content Disclaimer

This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither ProSoft Technology nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. Information in this document including illustrations, specifications and dimensions may contain technical inaccuracies or typographical errors. ProSoft Technology makes no warranty or representation as to its accuracy and assumes no liability for and reserves the right to correct such inaccuracies or errors at any time without notice. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of ProSoft Technology. All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components. When devices are used for applications with technical safety requirements, the relevant instructions must be followed. Failure to use ProSoft Technology software or approved software with our hardware products may result in injury, harm, or improper operating results. Failure to observe this information can result in injury or equipment damage.

Printed documentation is available for purchase. Contact ProSoft Technology for pricing and availability.

For professional users in the European Union
If you wish to discard electrical and electronic equipment (EEE), please contact your dealer or supplier for further information.

Warning – Cancer and Reproductive Harm – www.P65Warnings.ca.gov

Installation Instructions:

THIS EQUIPMENT IS AN OPEN-TYPE DEVICE AND IS MEANT TO BE INSTALLED IN AN ENCLOSURE SUITABLE FOR THE ENVIRONMENT SUCH THAT THE EQUIPMENT IS ONLY ACCESSIBLE WITH THE USE OF A TOOL.

SUITABLE FOR USE IN CLASS I, DIVISION 2, GROUPS A, B, C AND D HAZARDOUS LOCATIONS, OR NONHAZARDOUS LOCATIONS ONLY.

WARNING – EXPLOSION HAZARD – DO NOT DISCONNECT EQUIPMENT WHILE THE CIRCUIT IS LIVE OR UNLESS THE AREA IS KNOWN TO BE FREE OF IGNITABLE CONCENTRATIONS.

WARNING – EXPLOSION HAZARD – SUBSTITUTION OF ANY COMPONENT MAY IMPAIR SUITABILITY FOR CLASS I, DIVISION 2.

Agency Approvals and Certifications

Please visit our website: www.prosoft-technology.com

Table Of Contents

1 Start Here

This Quick Start Guide will help you:

- Setup your LDM Development Environment
- Build a sample ThingWorx application program
- Run the ThingWorx application program

1.1   Introduction

The ThingWorx-LDM-interface-lib is a software library (available for free at www.prosoft-tech-nology.com), to be used with a ThingWorx C SDK software library from PTC®. You can use these to create a sample application to allow communications between a CompactLogix PLC and a ThingWorx Server.

The sample application can be used as-is. This document provides step-by-step instructions on how to create the sample application. You can also extend the features of the sample application to suit your needs.

Our goal is to read and write data from the PLC to and from ThingWorx. This will be accomplished by running the sample application on the MVI69E-LDM, connected to the ThingWorks Server.
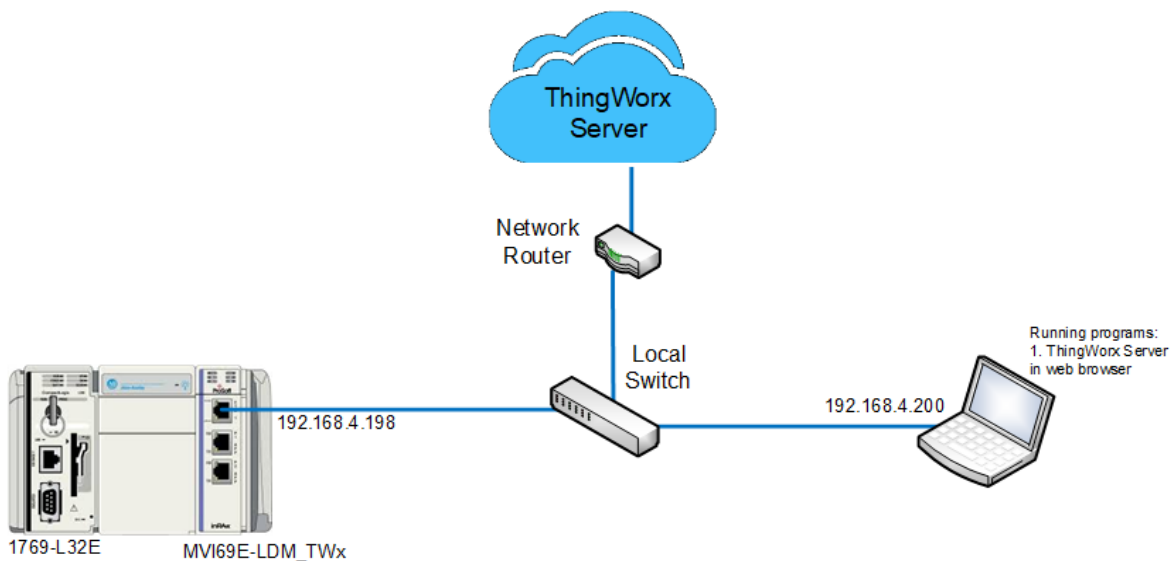
Figure: 1 Overall Architecture

The Quick Start Guide provides the steps to install the software onto a Windows 10 PC, build the sample application, and load it on the MVI69E-LDM. You will then configure it with a working sample.

Figure: 2 - Sample Application

1.2 Development Environment

The MVI69E-LDM development tools run in Linux. If you have experience with a previous ProSoft Technology LDM module, you may have setup a Linux Debian 6 Virtual Machine. The ThingWorx C library SDK will not compile on Debian 6, so this guide will step you through using a Docker® container on a Windows 10 PC.

2   Prerequisites

2.1   MVI69E-LDM-TW.Zip

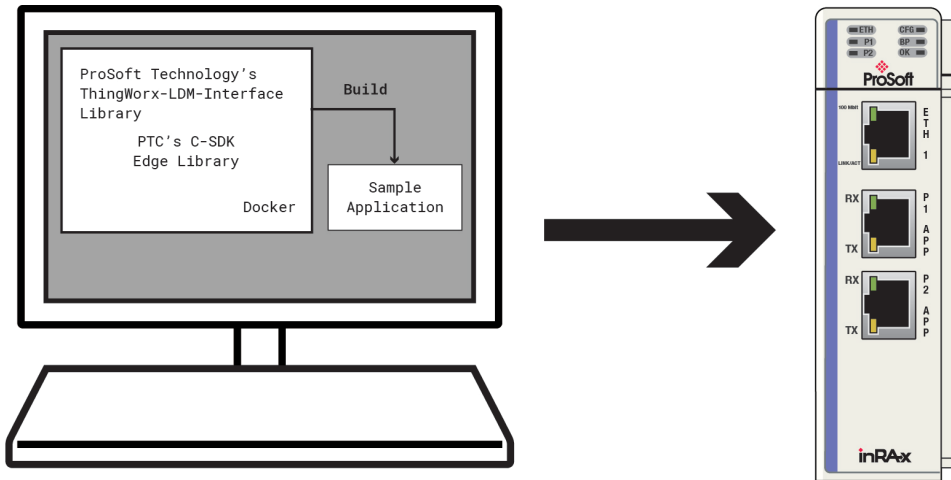Navigate to the MVI69E-LDM product page at www.prosoft-technology.com, and download the zip file **MVI69E-LDM-TW-xxx.zip** (where xxx is version number).

On your PC, create a folder C:\Workspace. Save and unzip this file in this folder. The interface library contains the following components:

c:\Workspace\	Subfolder	Description
cJSON		Files to overwrite the JSON files that come with the PTC C SDK library.
mvi69e-ldm		Source code of the sample application for MVI69E-LDM, and dependencies required to build it, and scripts to build firmware upgrade file.
ThingWorx-ldm-interface-lib		ThingWorx-LDM Interface Library
	build	Folder where target binaries are created during build
	docker	Toolchain to build source code and Docker configuration files to start container with build environment.
	entities	Specific files for ThingWorx sample
	scripts	Build scripts
	tw-ldm-interface-lib	Header files of the library
Throwing-ldm-sample-app-mvi69e	sample-app-mvi69e.sln	Option Visual Studio 2017 solution file for sample application

2.2   Source Code for the ThingWorx Edge C SDK

1. Download the C-SDK (**c-sdk-2-2-1-1321.zip**) file for the ThingWorx Platform from the PTC MarketplaceTM: https://marketplace.ptc.com/apps/193540/c-sdk#!overview.

2. Create a C:\Workspace folder on your PC, and copy the .zip file into it, and unzip it. It will be unzipped into a folder c-sdk-2.2.1.1321-master. Rename it to **ThingWorx-edge-c-sdk**.

3. You should now have C:\Workspace with these folders:

> **Note:** At the time of this writing, the C-SDK version was 2.2.1.1321
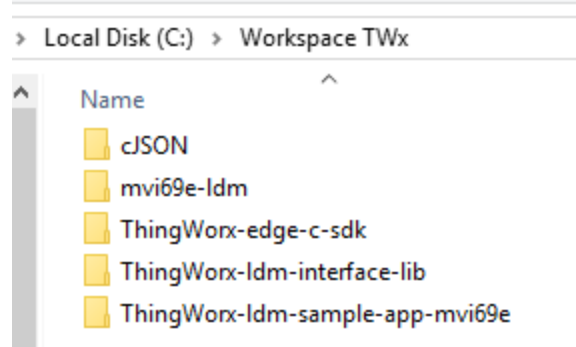
Figure: 3 Workspace example

2.3 ThingWorx Hosted Evaluation Server

Go to the ThingWorx Developer Portal and sign up for a free 30 day trial hosted Evaluation Server. The Developer Portal can be found here:

https://developer.thingworx.com/login?returnURL=%2Fen%2Fprofile%2Fhosting.

Figure: 4 Developer Portal Trial Sign-up

2.3.1 Launch the ThingWorx Server

1. Log in to the ThingWorx Evaluation Server.
2. Click the **CREATE SERVER** button. This will take a few minutes.
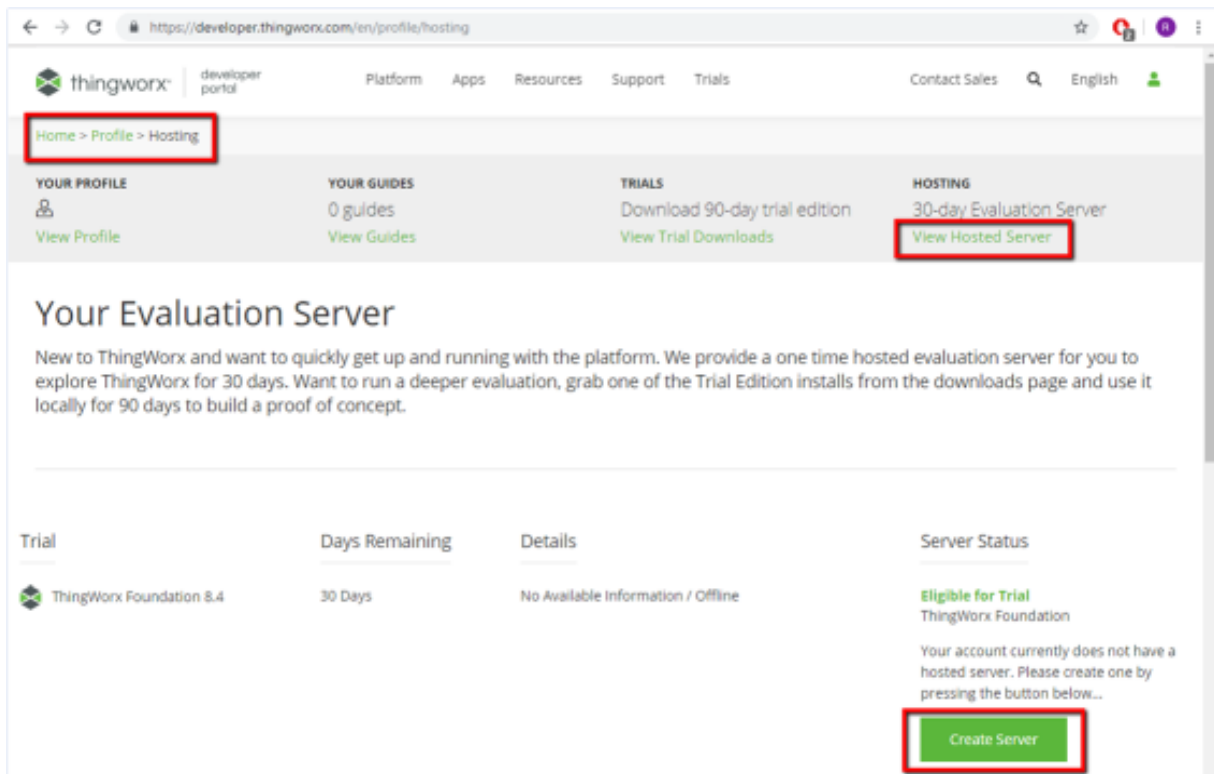
Figure: 5 Create Server

The Server Status will move to 'Provisioning' and then to 'Running'
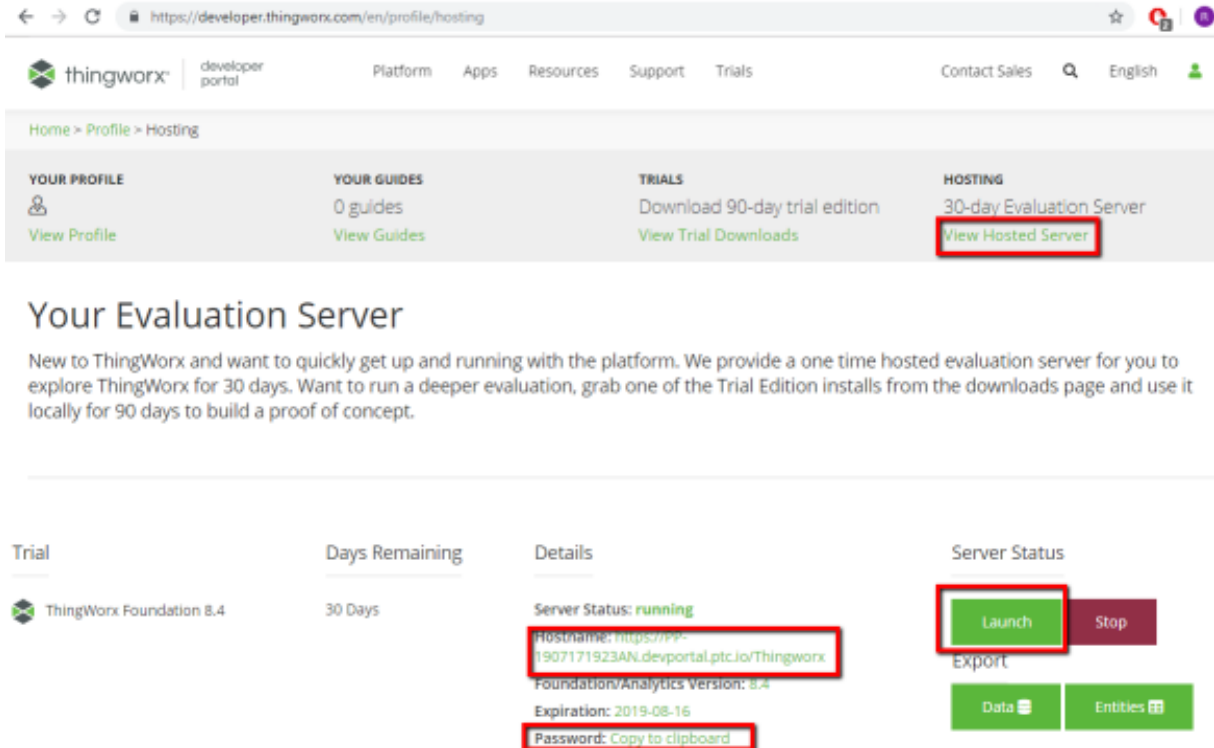
Figure: 6 Server Running

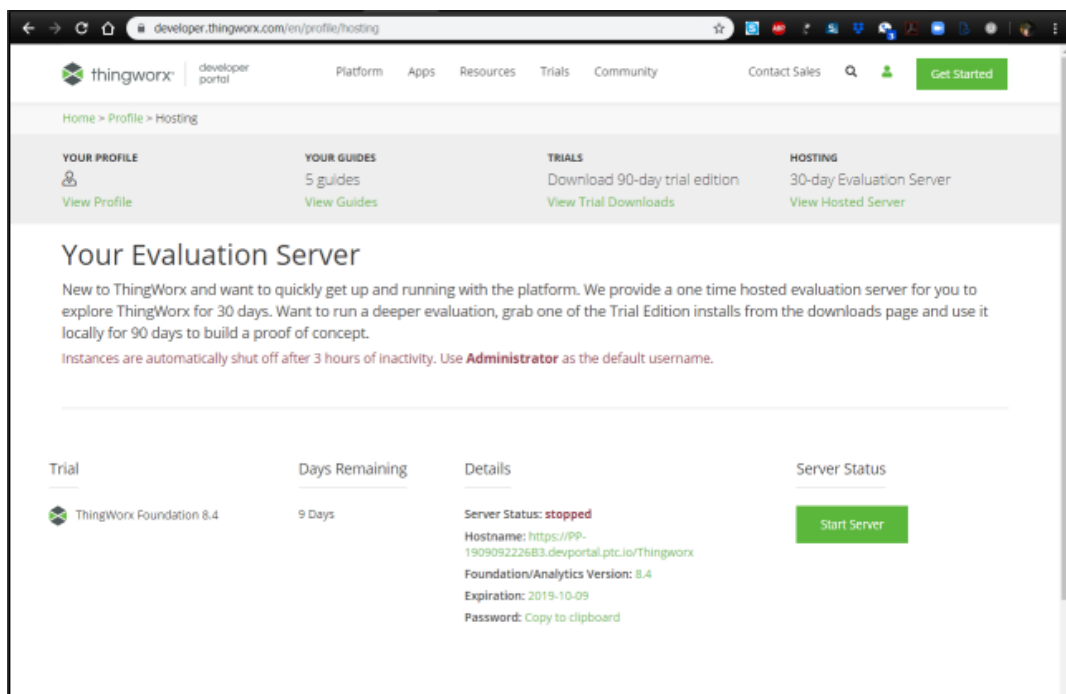1. Notice that the Password has an option to Copy to clipboard.

2. Click **COPY TO CLIPBOARD**, as it is required to login to the server. Paste it somewhere (such as Notepad) for later reference.

3. Make note of the Hostname.

4. When ready, click the **LAUNCH** button. It will option the ThingWorx GUI in Composer mode.

.

2.3.1   Get the App Key

Find the AppKey on the ThingWorx server's development page.

Figure: 7 ThingWorx Dev Page

Copy the Key ID. It will be used when configuring the sample application.
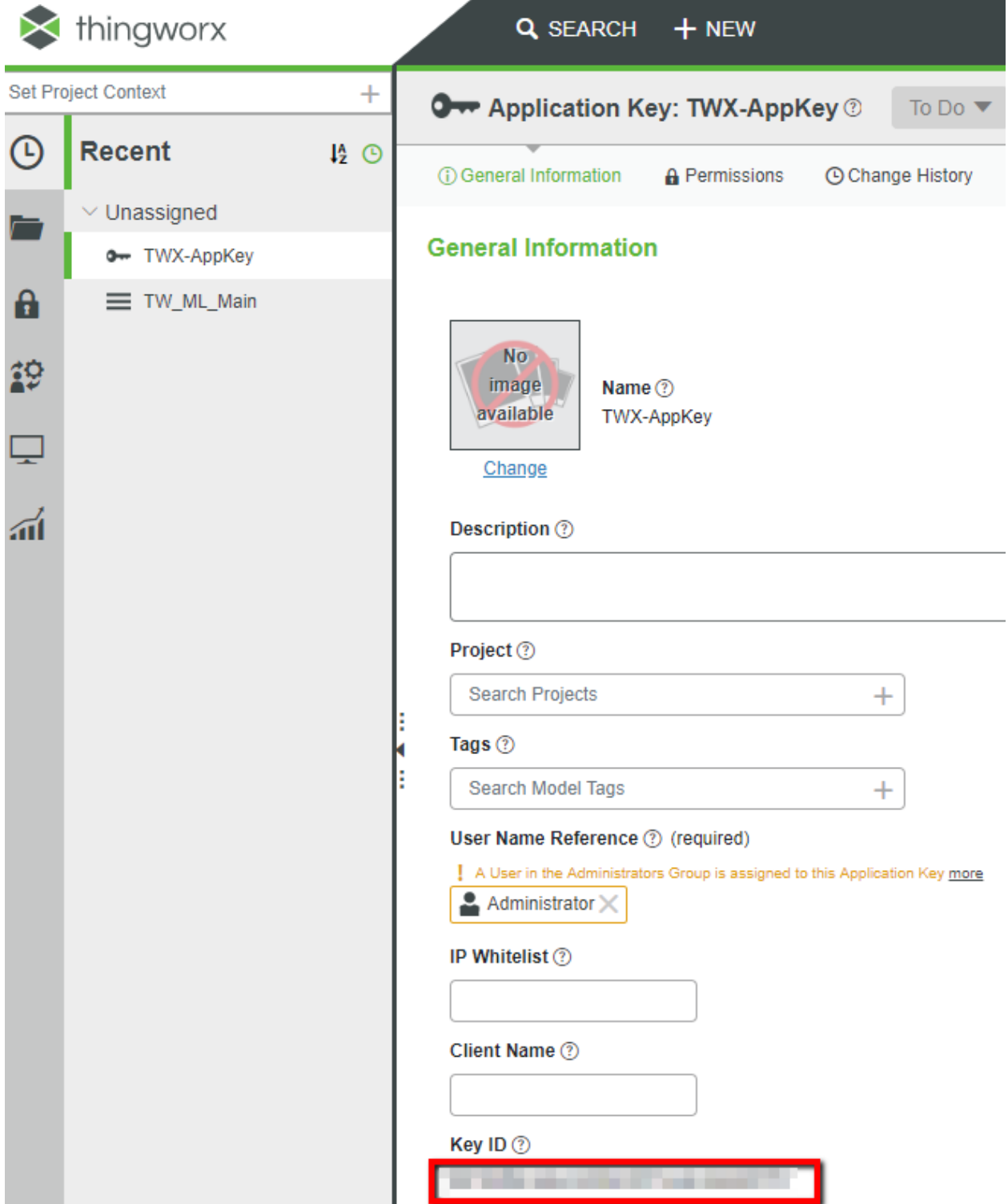
Figure: 8 Copy the Key ID

2.4　Turn on Hyper-V

Ensure that Hyper-V is turned on.

> Note that VMware can be used instead of Hyper-V, although Hyper-V is the recommended method.

2.5　Docker

Docker Desktop for Windows is required to run the toolchain from a container running Debian OS.

1. Find Docker Desktop for Windows here: https://www.docker.com/products/docker-desktop. Note that it should run in the default **Linux Containers** mode.

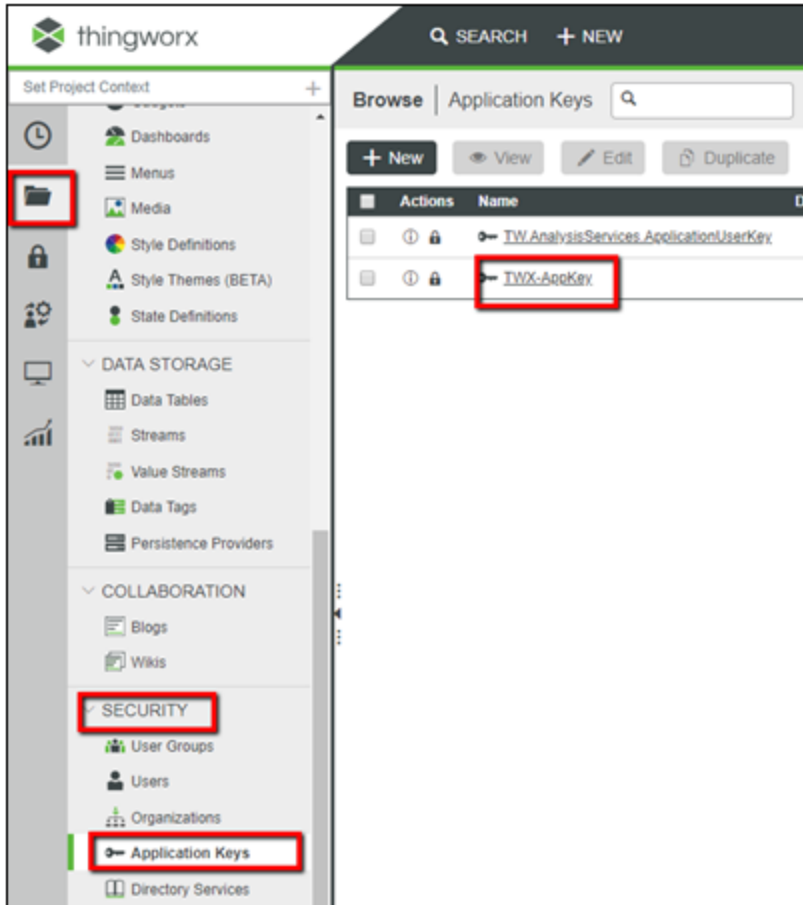2. PowerShell is already enabled on your Windows 10 PC. Ensure that it is enabled, as it is necessary for running Docker commands. Information on how to enable or install Power-Shell can be found here: https://docs.microsoft.com/en-us/-powershell/scripting/install/installing-windows-powershell?view=powershell-6

Note that container **psft-tw** will be left running after script completion. If you want to stop the container and remove it, you can modify script file build.ps1 (uncomment command docker container stop psft-tw at the end). SSH server will be running in the container, so it is possible to connect it using command from Windows console:

ssh user@localhost -p 4422

When asked for password, enter password.

The port number is 4422.

3 Development Environment Setup

3.1 Create User

Some Docker files will be stored in your Windows 10 \User folder. You can use your existing Windows login ID, or create a new one.

Also, the root folder of source code files (C:\Workspace) needs to be shared in order to access it from build container. In order to access this shared folder, Windows user credentials are required.

3.2 Shared Folder

Share the **C:\Workspace f**older.

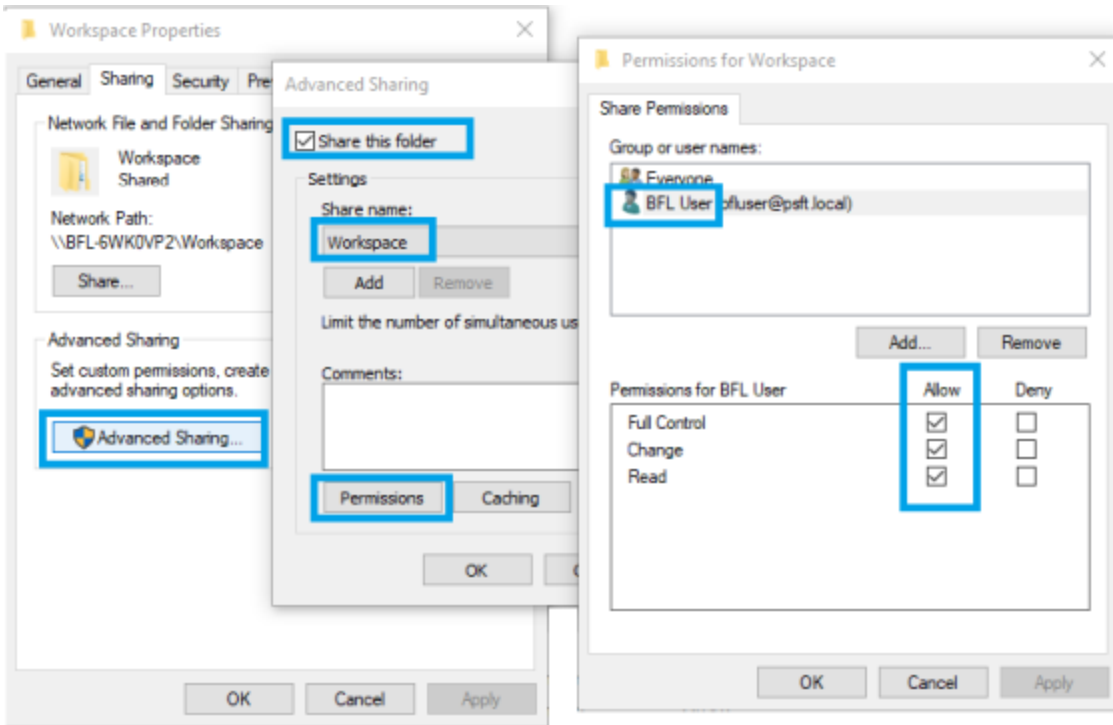Figure: 9 Shared Folder

3.3 Overwrite JSON Files

1. Navigate to **C:\Workspace\ThingWorx-edge-c-sdk\src\thirdParty\cJSON**.
2. Copy the 2 files from **C:\Workspace\cJSON** and paste them over the files in **C:\Workspace\ThingWorx-edge-c-sdk\src\thirdParty\cJSON**.
3. You can create a backup copy of the originals if desired (they are shown as _old in this picture):
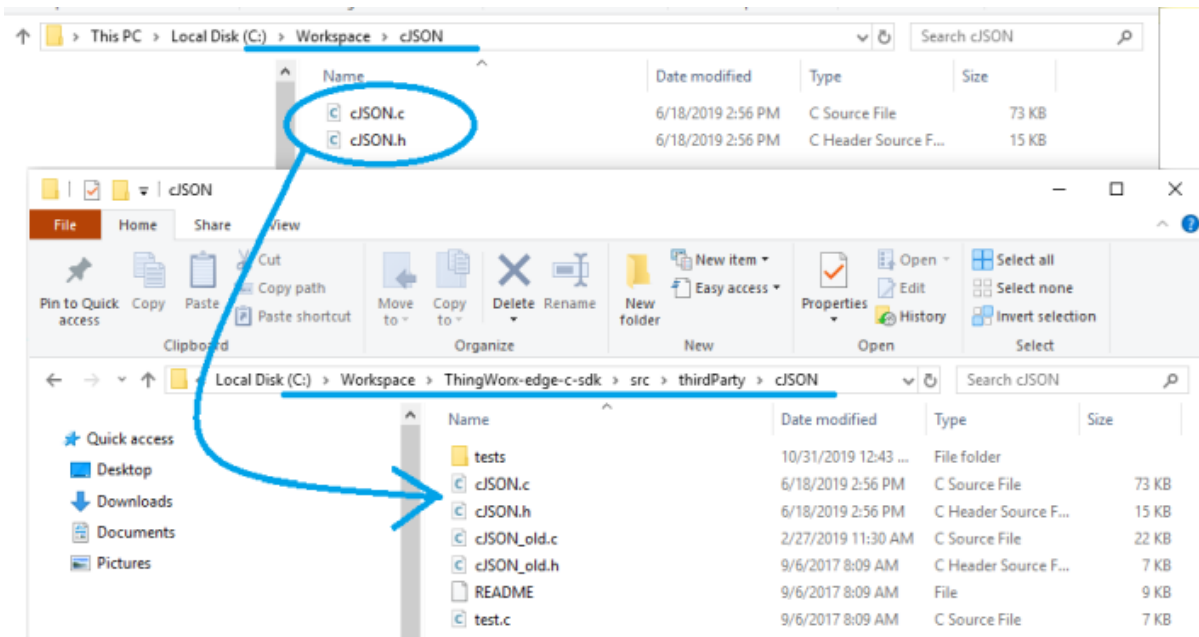
Figure: 10 Overwriting JSON files

4 Build

1. Acquire the IP address of your PC.
2. Open PowerShell console.
3. Navigate to **C:\Workspace\ThingWorx-ldm-interface-lib\scripts**.
4. Run script `build.ps1`:

```
./build.ps1 -SHARED_FOLDER //192.168.1.73/Workspace

-SHARED_FOLDER_USER bfluser -SHARED_FOLDER_PASSWORD passwd
```

- Replace the *IP address* shown (192.168.1.73) with the PC's IP address.
- Replace the *user id* shown (bfluser) with your user ID, from step 3.1
- Replace the *password* shown (passwd) with the password for your user ID

As result, a Debian 9 image should be polled from the Docker Hub, required components installed to it, including toolchain, source code projects built, and firmware image created:

C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\firmware\mvi69e-ldm.firmware_<version number>_<date>.firmware

5 Connect to the MVI69E-LDM

5.1 Physical Connections

1. With the MVI69E-LDM in the CompactLogix rack, connect the top Ethernet port to your local network, and establish a connection to the PC.

2. Connect the PC via a USB to Serial adapter, to the 2nd (middle) port in the MVI69E-LDM module. This is for debugging purposes using TeraTerm (Open Source)
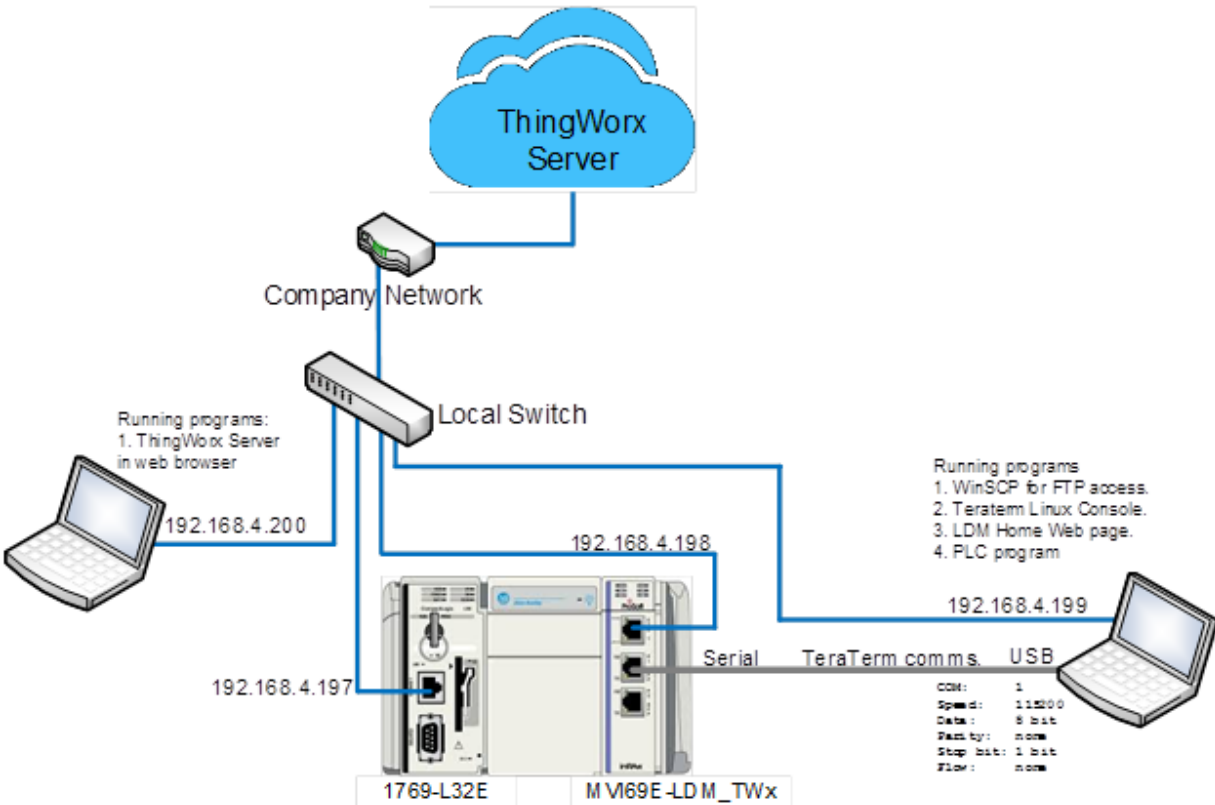
Figure: 11 Cabling Configuration

5.2 Configuring the CompactLogix Processor

1. Open the MVI69E_LDM_TWx.ACD program and change the appropriate chassis type to match your hardware and firmware.

2. Download MVI69_LDM_TWx.ACD file to the CompactLogix processor by choosing **Communications > Who Active > Download**.

6   Run the Sample Application

6.1   Check the Certificate Dates

1. In **C:\Workspace\ThingWorx-ldm-sample-app-mvi69e**, double-click on the *root_ca.cer* file and confirm that the current date is valid.
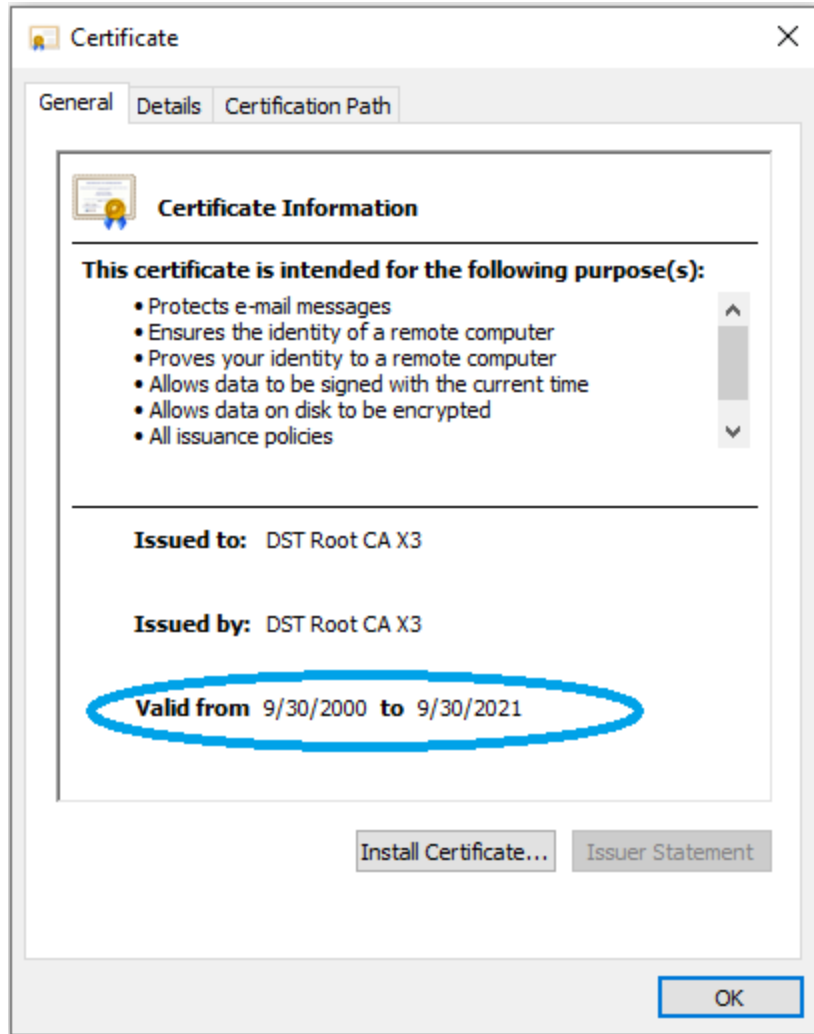
Figure: 12 Certificate Information

If a certificate date validation error occurs, check the date on the module. The module's date can be adjusted by hand in Linux, or done programmatically. Review the set-time and S88-tw files. The date can also be set on the module's webpage, but by that time it is too late, as the startup sequence has already occurred.

2. If the hosted ThingWorx CA certificate is changed by PTC, or if it expires, or if you are using a local ThingWorx server, then install the ThingWorx server certificate's root CA certificate.

6.2 Install the Sample Application

> **Note:** If an application is currently running on the MVI69E-LDM, back it up before proceeding.

1. Download the *.firmware* file that was just built to the module via the module's webpage. Refer to MVI69E-LDM Developer Manual for details.

2. At the end of the *Firmware Update* process, the module will reboot.
3. The sample application will start running automatically after reboot. It still needs to be configured. Continue to the next section.

5.1 Set the MVI69E-LDM's Ethernet IP Addess

Set the MVI69E-LDM's Ethernet port IP addresses by modifying the `/etc/net-work/interfaces` file on the module. Refer to the Developer Manual for detailed information about the interfaces file.
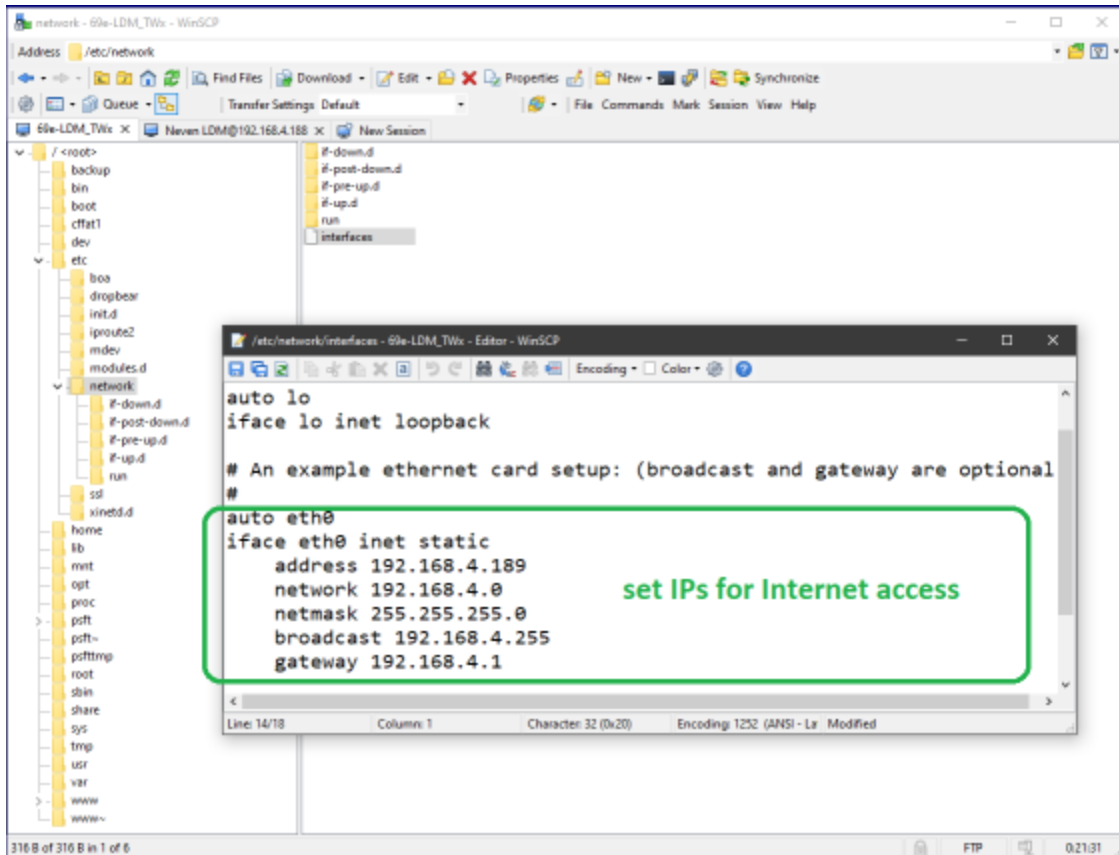
Figure: 13 Setting the Ethernet IP address

5.2 Configuration of the Sample Application

The configuration file **config.json**, in JSON format, should be edited manually (access it over FTP using a client such as WinSCPTM).

There are only a few changes needed for the sample application.

5.2.1 Set the Host

Set the IP address of the Host ThingWorx server. From the PC, open a command prompt and ping the ThingWorx hosted server URL. The server's IP address will be used for this parameter.

5.2.2 Set the App Key

Set the AppKey of the ThingWorx server.

When using the hosted ThingWorx evaluation server, no additional configuration changes are necessary.

5.2.3 Structure of the Configuration File

This is a description of the elements of the configuration file.

TwServer

Settings required to connect to the ThingWorx server:

- Host – Defines host name or IP address of the server;

If the module is not connected to the Internet where ThingWorx server runs, this can be the IP address of the PC in the local network where forwarding of local traffic to the Internet is configured using port-forwarding.

- **Port** – Port number.
- **Timeout** – Connection timeout in ms.
- **UseHttps** – Optional parameter. If set to '0', directs to use http protocol. If non-zero, use https protocol. Helpful in cases when port number is not standard port used by http or https.
- **DisableCertificateValidation** – If set to a non-zero value, eliminates certificate validation. Useful to enable connections to server with self-signed certificate or with a certificate which is issued by root CA certificate not set in *root_ca.cer* file.
- **RootCaFileName** – Path to file with root CA certificate in Base-64 encoded PEM format. Default file name is *root_ca.cer*, located in the same folder as sample application.
- **ConnectionRetryCount** – Number of connection retries. It is recommended to leave default value 1.
- **AppKey** – Application key. Specific for each instance of the ThingWorx server.

Things

A list of things defined in the ThingWorx server. The sample configuration has one thing – SteamSensor. Each thing has a list of properties, property name, and data type.

Thing name and property names are used in the Tags section of the configuration file.

To facilitate the possibility of detecting the current state of connection from the LDM module to the PLC, each thing might have an additional property not listed in its Properties node. If the thing definition in configuration file has a node **IsConnectedPropertyName**, then it will have a property of BOOLEAN data type with name equal to the value of this node. By default, node **IsConnectedPropertyName** is set to value **IsPlcConnected**, so each thing will have property with name **IsPlcConnected**.

PlcPath

Defines connection string defined to connect to the PLC.

SyncTimeWithPlc

A flag indicating if system time should be synchronized with the PLC.

0 - Default value. Synchronize once, if current system year is less than 2019, which is usually the case after system restart.

1 - Synchronize one time after first successful connection to the PLC.

2 - Synchronize after every successful connection to the PLC.

> **Note:** Synchronization is performed only if current year in PLC is greater or equal to 2019.
> This prevents the re-setting of system time when PLC time is not set.

Tags

Defines the list of tags defined in the PLC, and settings to map them to the ThingWorx things.

- **Offset** - Offset of the tag in PCL address space, in WORDs. Defined only for MVI69E.
- **DataType** - Data type of the tag in the PLC. Possible values are BOOL, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, LDWORD, STRING82.
- **ScanRate** - Defines how often the tag value is read from the PLC. Should be the multiple of lowest scan rate defined for all tags mapped to the same Thing.
- **Access** - Defines if the tag is read-only (value RD), or writable (value RDWR)
- **Thing** - Name of the corresponding thing in ThingWorx. Case sensitive.
- **Property** - Name of the property defined in ThingWorx. Case sensitive.

StatusPrintIntervalInSeconds

The main thread of the sample application checks the connection status with ThingWorx servers, and logs that information with an interval defined with this optional parameter. The default value is 10 (seconds).

After changing of configuration file, the process tw-ldm-sample-app-mvi69e should be re-started. This can be done via telnet terminal, or by rebooting the module (the latter option requires existence of the script `/etc/init.d/S88-tw`, which is installed by Firmware Update).

5.3 Running the Sample Application

The sample application was installed in the section Install the Sample Application. With the configuration completed, restart the sample application.

To restart the application, reboot the module. Or from the module's `/psft/sample/tw` directory, locate the sample application's process ID with the top command, and then kill that process ID. Then, start up the sample application by running the command:

```
./tw-ldm-sample-app-mvi69e
```

Now that the MVI69E-LDM is running the sample application, the next step is to configure the ThingWorx server.

5.4 Import Sample Entities

Log in to the ThingWorx Evaluation Server that you created in section ThingWorx Hosted Evaluation Server.

Entities are the model definition of the user's data that will be created in the ThingWorx Server instance. The ThingWorx C Edge SDK that was previously downloaded includes a SteamSensor XML file.

1. In the ThingWorx server, click IMPORT/EXPORT, select Import, and Browse to **C:\Workspace\ThingWorx-edge-c-sdk\examples\SteamSensor\import**.
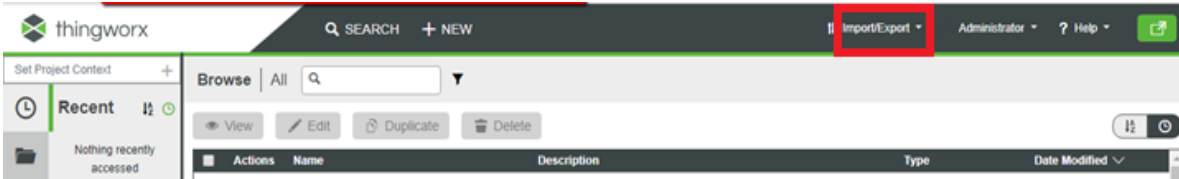
Figure: 14 Import

2. Select the file **Entities_SteamSensor.xml.**

3. Click **OPEN**. Upon successful import, click **CLOSE**.

4. Click on **BROWSE** in ThingWorx Composer, and select the *Things* section. Notice the new thing SteamSensor appeared.
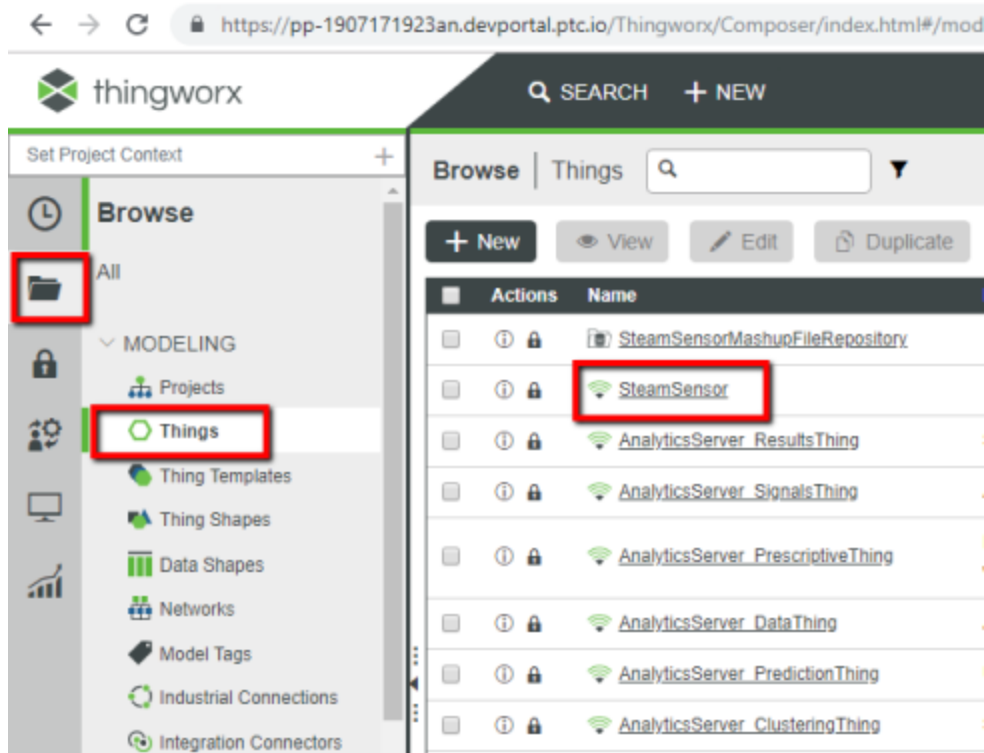
Figure: 15 Steam Sensor

5. Expand the *Mashups* section to see the *SteamSensor* mashup in the list of mashups. Mashup is a graphical screen design unit in ThingWorx.
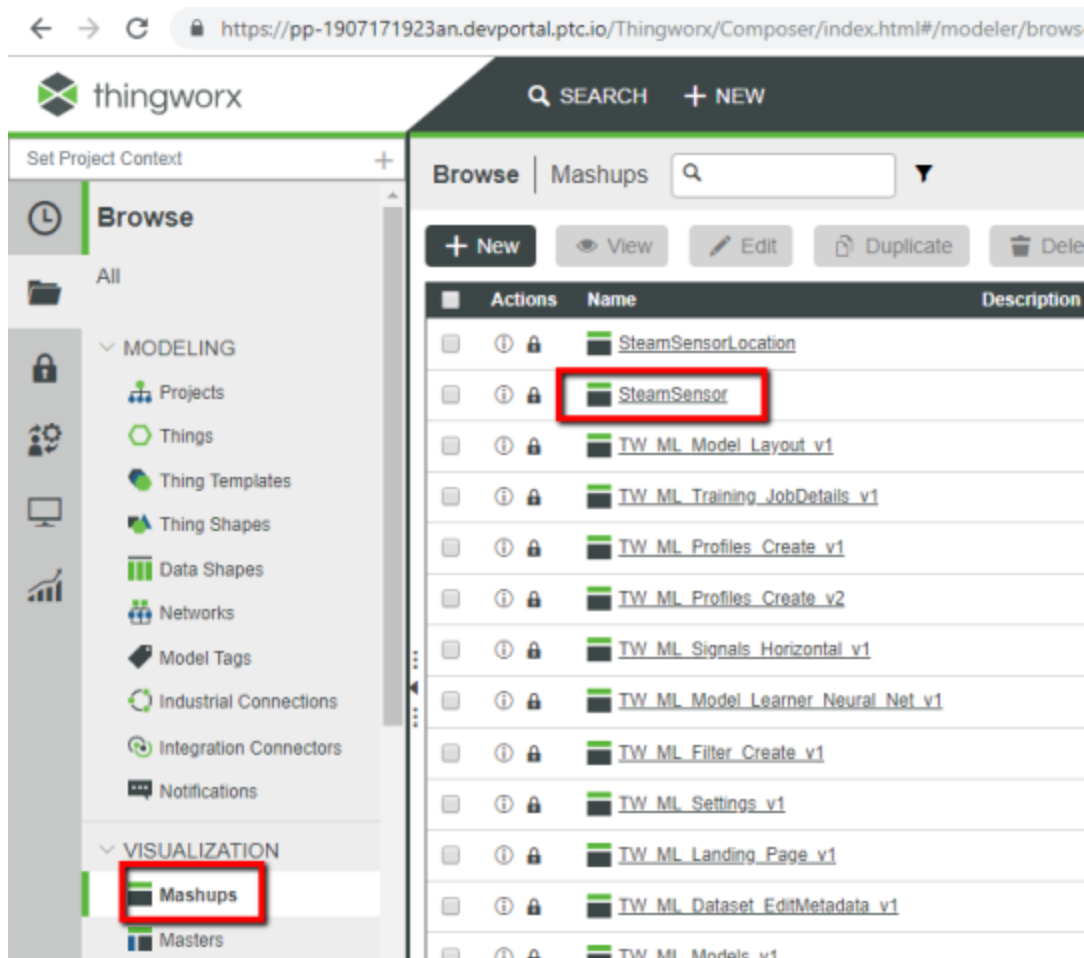
Figure: 16 Mashups

6. Click the *SteamSensor* mashup to open the designer page:
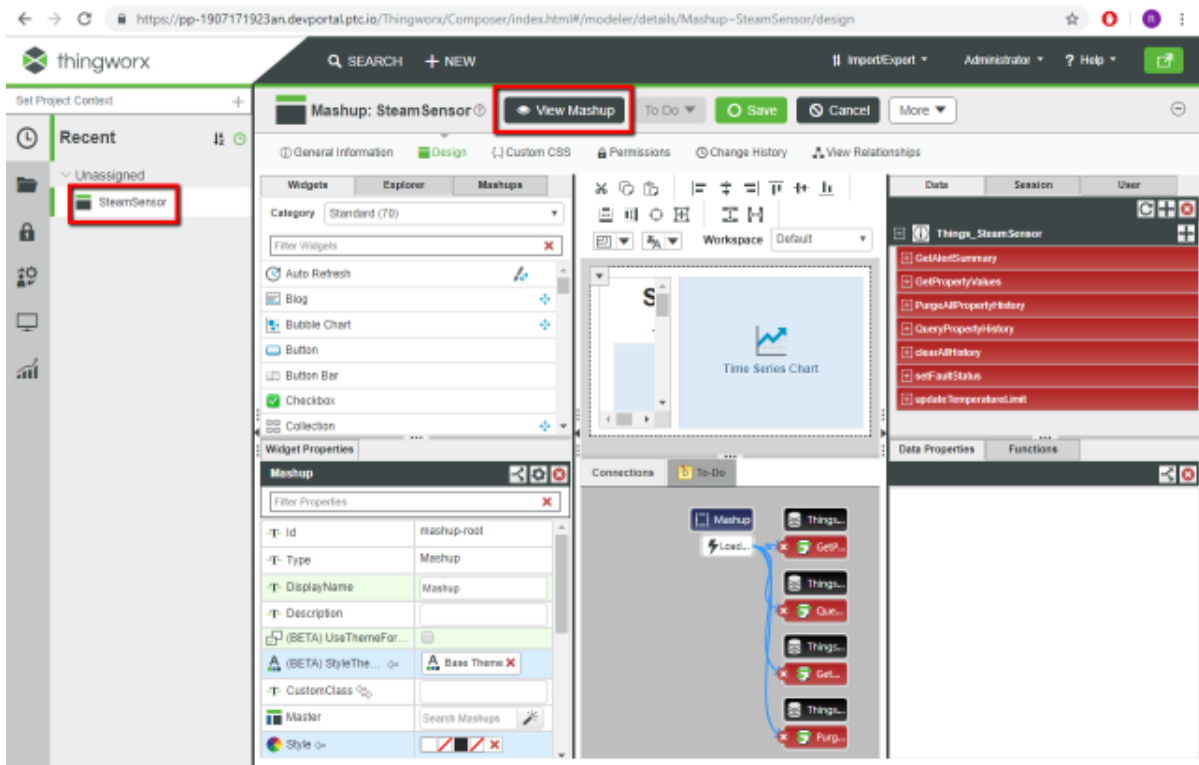
Figure: 17 Designer Page

The mashup shows the GUI widgets in the center panel, with their properties and data binding settings displayed in other panels. From here, click on the **VIEW MASHUP** button to view it in runtime mode.

In the runtime view, the Connected label will display with a green checkmark , indicating the ThingWorx server is exchanging data with the PLC.

Figure: 18 Runtime View

7 Installing Root CA Certificate

If you are using the ThingWorx hosted server, and the current date is within the Valid date range on the certificate, then no additional certificate steps are required.

In order to connect from sample application to the ThingWorx server via https protocol, it should be configured to trust to the root certificate of the SSL (https) certificate of the server.

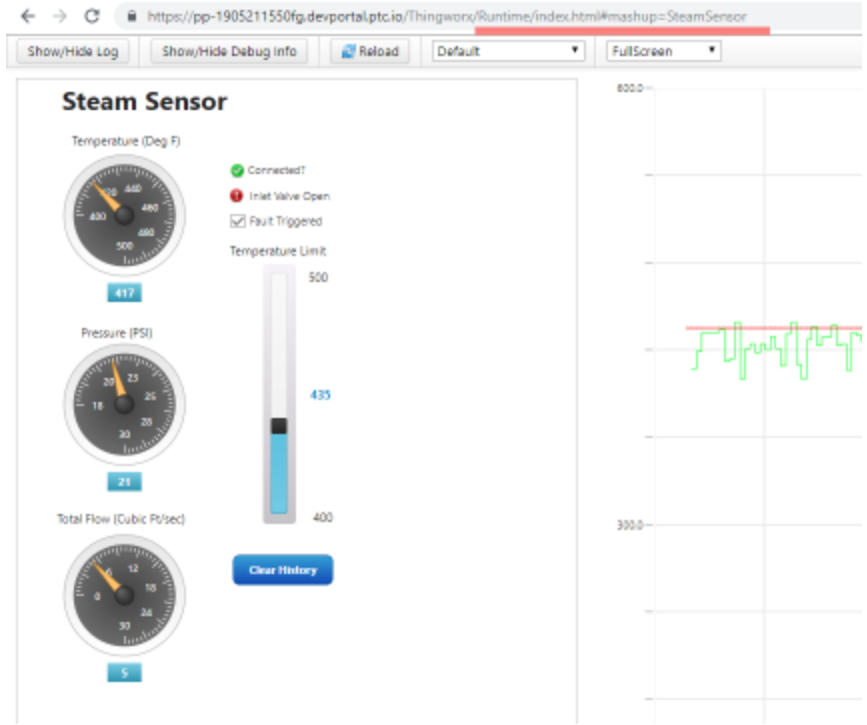The certificate (**root_ca.cer** file) copied to the MVI69E-LDM belongs to Digital Signature Trust Co®, a well-known Certificate Authority, which is a parent of SSL certificates used in ThingWorx trial servers, hosted by PTC. Copying this certificate to the MVI69E-LDM makes sure that the sample application is trusted by the PTC servers. No additional certificate steps are required.

It is worth mentioning that servers hosted by PTC have their https certificate issued by free certificates provider **Let's Encrypt ®** (https://letsencrypt.org/). Therefore, if you get a certificate for local ThingWorx installation from **Let's Encrypt**, it will be accepted by the sample application without additional configuration changes.

7.1 Local Server Instance

If you decide to use a ThingWorx local server instead of the hosted server, there additional certification steps are required.

In case of using local instance of ThingWorx server with SSL certificate, which is issued by root CA certificate other than Digital Signature Trust Co, that root CA certificate should be saved in file and copied into the module following instructions:

1. Connect to your ThingWorx server instance from a web browser and open its certificate. In Google Chrome, this can be done by clicking on the icon with a lock icon in front of the URL. When the menu opens, select the **Certificate** command:
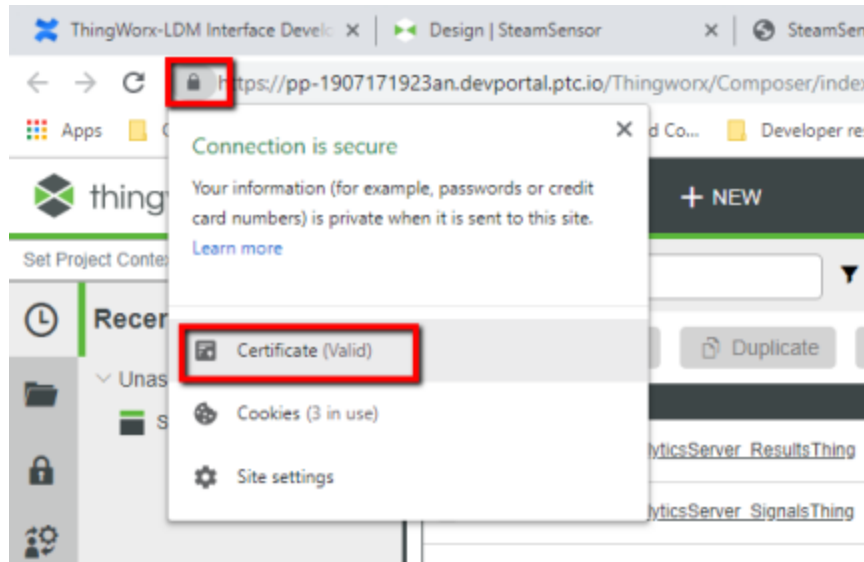
Figure: 19 Open your ThingWorx Certificate

2. As a result, the SSL certificate will be displayed in new window. Click on *Certificate Path* tab to display the certificates chain:

Figure: 20 Certificates chain

3. Select root level certificate entry from the Certificate path tree (in the picture below **DST Root CA X3**), and click on **VIEW CERTIFICATE** button:
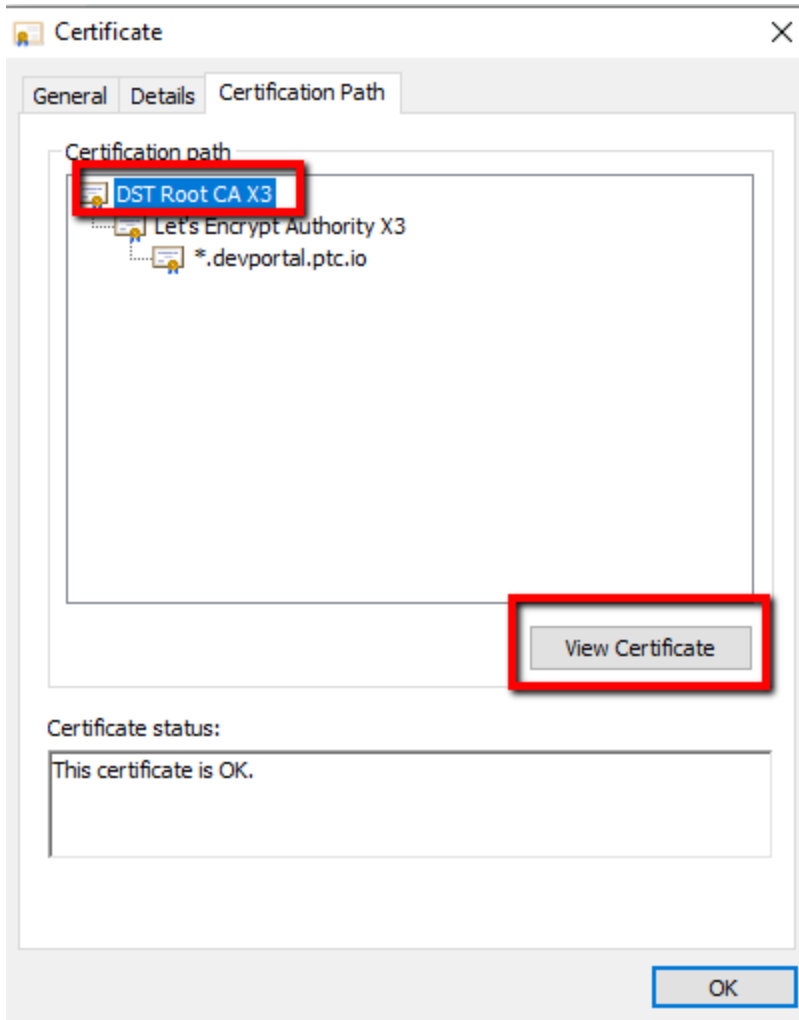
Figure: 21 View Certificates

4. In new window, select Details tab page and click on the **COPY TO FILE …** button.
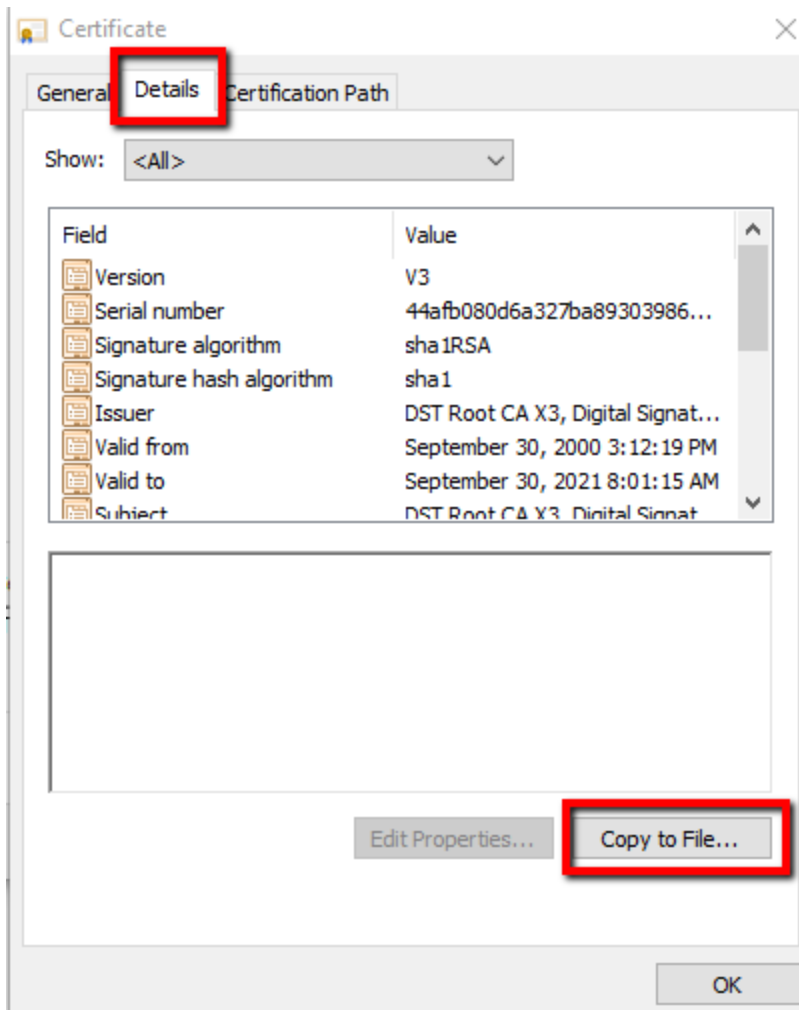
Figure: 22 View Certificate Details

5. The *Certificate Export Wizard* window will be opened. Click on the **NEXT** button.
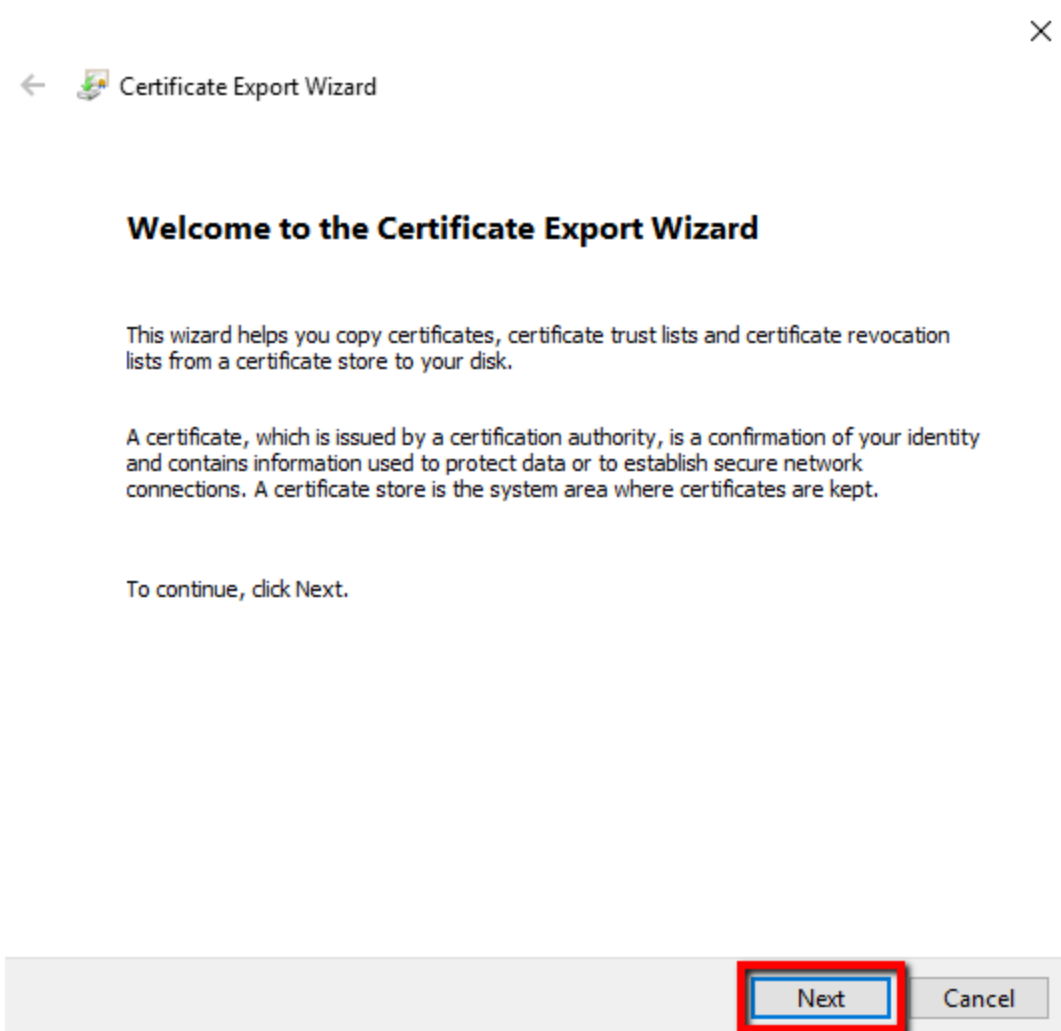
Figure: 23 Certificate Export Wizard

6. Select format **Base-64encoded X.509 (.CER)** and click on the **NEXT** button:
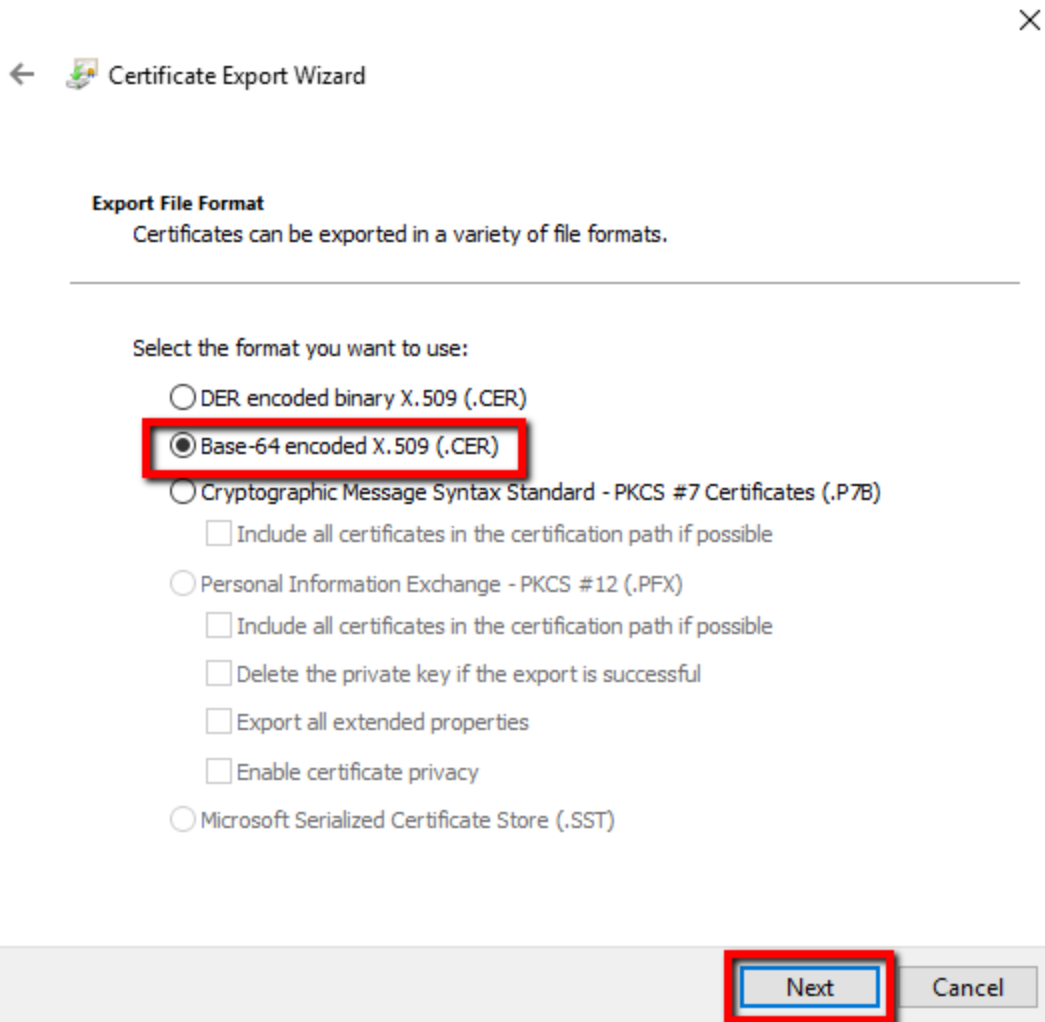
Figure: 24 Selecting Base-64 encoded .509

7. Select location and file name to export, by clicking on the **BROWSE** button:
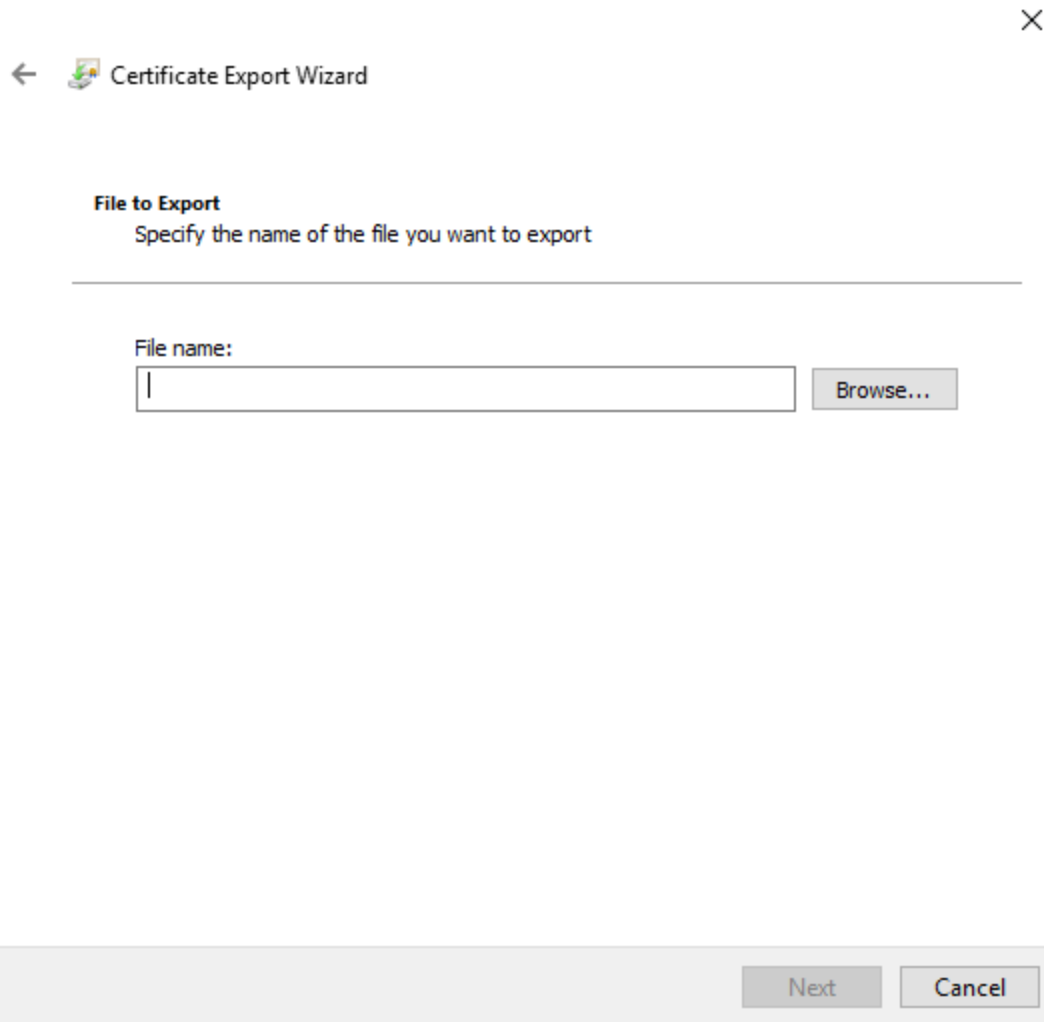
Figure: 25 Selecting Filename to Export

8. Export to file, and name it **root_ca.cer** (default name). You can choose a different name if you prefer, and in that case modify the configuration file **config.json** to reference the different name.
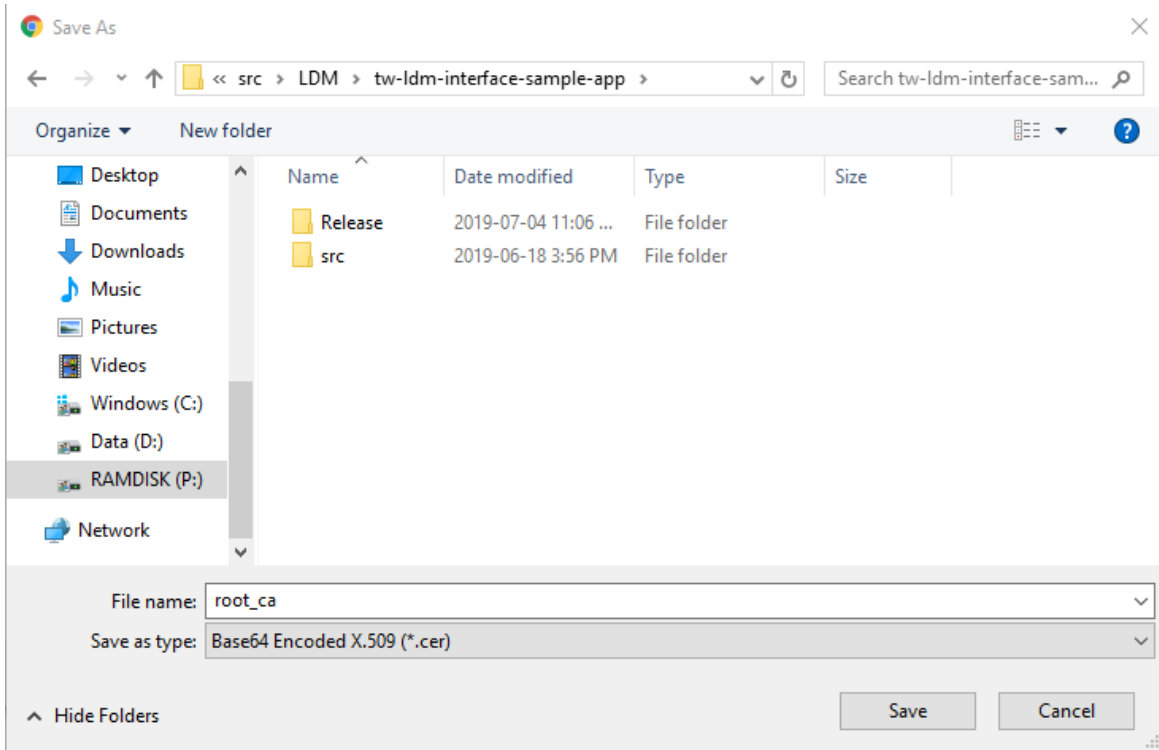
Figure: 26 Exporting the File

9. Click on the **FINISH** button to complete export.

10. Once saved, the certificate needs to be copied to the module, in the folder /psft/sample/tw (for example, using WinSCP). If the certificate was saved into the file with name different than default **root_ca.cer**, then **config.json** file needs also to be edited to point to that new file name.

8 ThingWorx-LDM Interface Library

This section is intended for developers building custom applications using the library. It describes high-level design of the library and main API functions required for customer applications.

8.1 Component Diagram

Interaction between components is shown in the following diagram:

ThingWorx Server

Unit test project.
Can be built for different targets.

Windows (has Visual Studio project)

Debian 9.5 (VM with toolchain)

MVI69E-LDM Module

Sample application tw-ldm-interface-sample-app

ThingWorkx Edge C SDK Library

https

Initialize

Callback

tw-ldm-interface-tests Executable , automated unit tests

ThingWorx-ldm-interface-lib ibrary

Init/Start/Stop

mvi69e-datm3

tw-ldm-interface-sample-app

Executable

Dependencies for sample application.

JSON Configuration File

Example project, intended to be used by end customers as a base to start. Can be used as a starting point.

PLC

Figure: 27 Component Diagram

8.2   Main API Functions and Data Flow

8.2.1   Functions Implemented by the Library

Below are functions defined in header file **C:\Workspace\ThingWorx-ldm-interface-lib\tw-ldm-interface-lib\inc\tw-ldm-interface-lib.h** are expected to be called from the user application:

int tw_ldm_initialize(const char* path_to_config_file, get_app_key_callback_function_type app_key_cb_function);

This function should be called at application start-up once, to initialize ThingWorx SDK internal structures and start its threads.

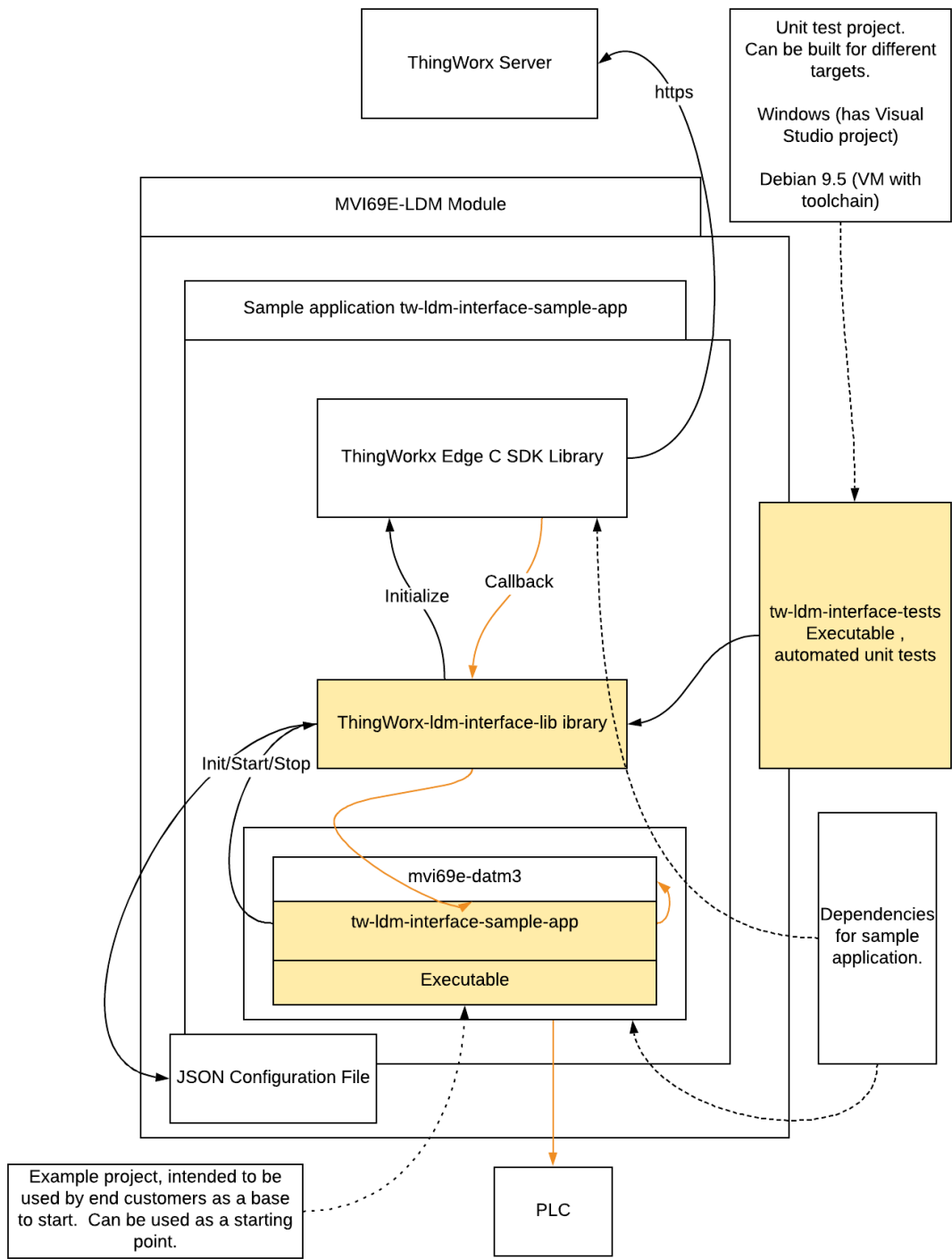As arguments, it takes path to the JSON configuration file, and optionally pointer to the function to get application key used to connect to ThingWorx Server.

In case of success, function returns 0.

int tw_ldm_connect(void); - This function connects to the ThingWorx server. In case of success return 0.

int tw_ldm_disconnect(void); - Disconnect from ThingWorx server.

int tw_ldm_is_connected(void); - Returns non-zero value if currently connection to ThingWorx server is established.

int tw_ldm_is_connecting(void); - Returns non-zero value if currently connection to the ThingWorx server is in progress.

int tw_ldm_clean(void); - Frees resources used by the ThingWorx-LDM Interface library and by the ThingWorx C Edge SDK. Should be called before application exits.

void tw_ldm_log(enum tw_ldm_log_level log_level, const char *format, ...);

Can be used by the user application to log messages into log file.

Passed arguments are log level, C-style format string and optional data to log.

8.2.2   Callback Function Declarations

Functions - Callbacks, which are called by ThingWorx-LDM Interface Library during runtime and should be implemented by the user application (default implementations are provided in sample application):

int tw_ldm_read_value(tw_tag* tag, plc_value* value);

- Called to read value of a tag from underlying PLC.
- Input argument tag carries information about tag which is being read.
- It is expected that implemented function returns read value in output argument value in case of success and returns 0

int tw_ldm_write_value(tw_tag* tag, plc_value* value);

- Called to write new value to a tag.

int tw_ldm_is_connected_to_plc(void);

- If the module is connected to the PLC, then it returns 1. Otherwise, returns 0.
- This function is called by the ThingWorx-LDM library for each Thing when it is polled, if for the thing in the configuration file field IsConnectedPropertyName is set to name of the property used to detect state of the PLC connection. If this field omitted or set to empty value, then this function will not be called.

int tw_ldm_get_status(char is_verbose, char** buffer, uint16_t max_size);

- Used to get information on current status of communication with the underlying PLC.
- If input argument **is_verbose** is not 0, then more detailed information is returned. If it is 0, then brief information is returned.
- Result is copied into provided by the argument **buffer** memory buffer. It might be pre-allocated by the caller. In this case its size is passed in **max_size** argument. If buffer points to NULL, then **max_size** still can be used to limit size of returned text.
- Function is expected to return 0 in case of success.
- When argument **is_verbose** has non-zero value, sample application **tw-ldm-interface-sample-app** returns content of log file in the buffer.

Examples of using of those functions can be found in source code of the sample application. Particularly, function tw_ldm_get_status is mapped to the ThingWorx SteamSensor thing's GetStatus service. It is called when in the SteamSensor mashup, value in the field Get Status is edited: it is passed as input argument, and returned in the buffer output argument value is displayed in the text area field following below Get Status field.

Note that logging feature uses standard Linux daemon syslog. The script S10-syslog, which configures and starts it is located in source code folder of the sample application, which is included into the firmware and installed in the module into folder **/etc/init.d** and starts syslog daemon at system boot. It is configured to log messages into file **/www/html/log/messages.txt**. Therefore it can be accessed via module's web server using address **http://192.168.0.250/log/messages.txt**.

It is configured to limit file size to 32 Kbytes, and maximum number of files set to 1. With these settings, when log file size exceeds 32Kb, it archives currently active log file into file messages.txt.0, and continues logging in new file **mssages.txt**. Archived copy can be accessed via URL **http://192.168.0.250/log/messages.txt.0**

8.3   Data Flow for Reading of Tag Values

Orange lines in the component diagram above illustrate data flow channel. At initialization stage, the **Thingworx-ldm-interface-lib** registers callback functions at the ThingWorx C Edge SDK. Note that callback function to read values is then called one time per each Thing, with the rate equal to smallest scan rate of all tags mapped to that Thing's properties.

For example, for **SteamSensor** thing in the sample application there are multiple tags mapped, some of them have scan rate 500 ms (Temperature_tag), some 1000 (Pressure_tag), etc. The minimum rate is 500 ms, so the callback function from ThingWorx C Edge SDK to the Thingworx-ldm-interface-lib is called with that rate 500 ms. Then **Thingworx-ldm-interface-lib** calls **tw_**

ldm_read_value implemented by the user application (in the component diagram - tw-ldm-interface-sample-app), for each tag, with a rate depending on the scan rate of the tag.

For example, in the **config.json** file in the sample application scan rate for tag Temperature_tag is 500 ms, and for the Pressure_tag - 1000 ms. The rate for the whole **SteamSensor** thing is 500 ms, for the Temperature_tag function tw_ldm_read_value will be called every 500 ms, and for the Pressure_tag - every 1000 ms.

Note that in this example, the greater scan rate is a multiple of the smallest rate (1000 = 500 x 2). If the rate for Pressure_tag was defined as 1200, then it would be updated every 500 * 3 = 1500 ms, i.e. closest greater multiple of 500.

8.4   Data Flow for Writing of Tag Values

To write values, the **Thingworx-ldm-interface-lib** library registers 2 Thing services: **SetNumericProperty** and **SetStringProperty**.

They both take 2 arguments: **name** (for thing's property name), and value - to pass value. The service **SetNumericProperty** is intended to write numeric values as well as Boolean type values. They are converted by the **Thingworx-ldm-interface-lib** to data type of PLC tags.

The screenshot below shows **SetNumericProperty** service for the **SteamSensor** thing in ThingWorx composer. You can supply values for arguments and call it by clicking on the **EXECUTE** button.
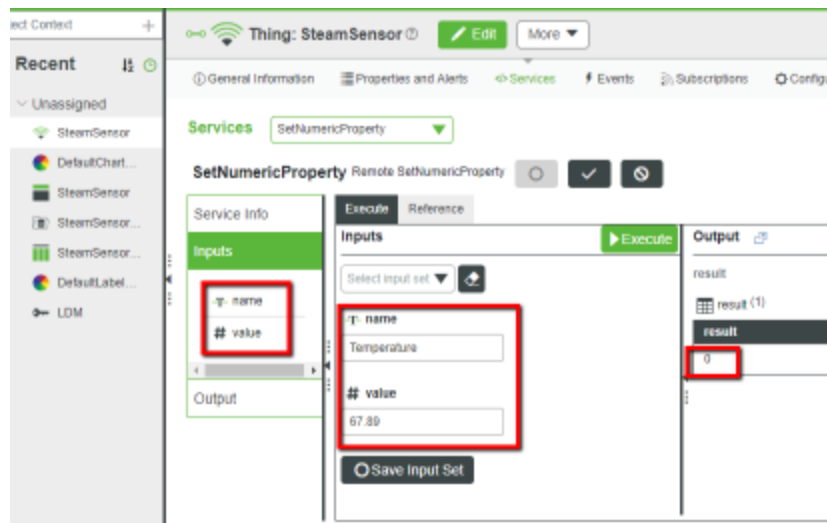
Figure: 28 SetNumericProperty service

In the composer, to write values for properties, a separate local service can be created for each property. This calls the remote **SetNumericProperty** or **SetStringProperty** service.

For example, the screenshot below shows a definition of the local service **SetPressure**. You can see that it is implemented as JavaScript code, which sets argument values and calls service **SetNumericProperty**:
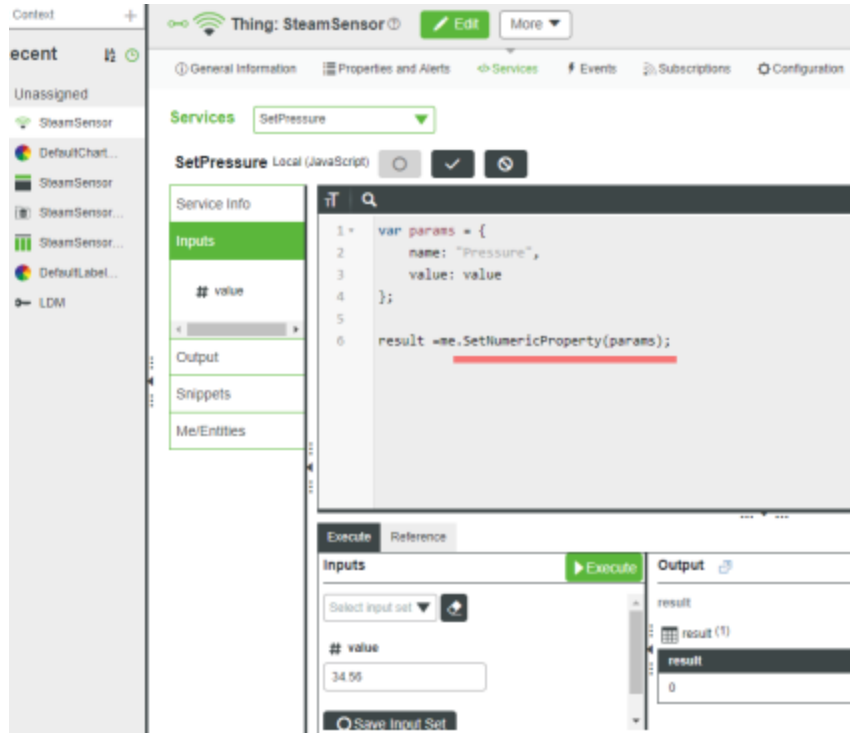
Figure: 29 Local Service Definition

For the numeric entry widget, where value to write is entered, binding can be setup to call this local service **SetPressure**:
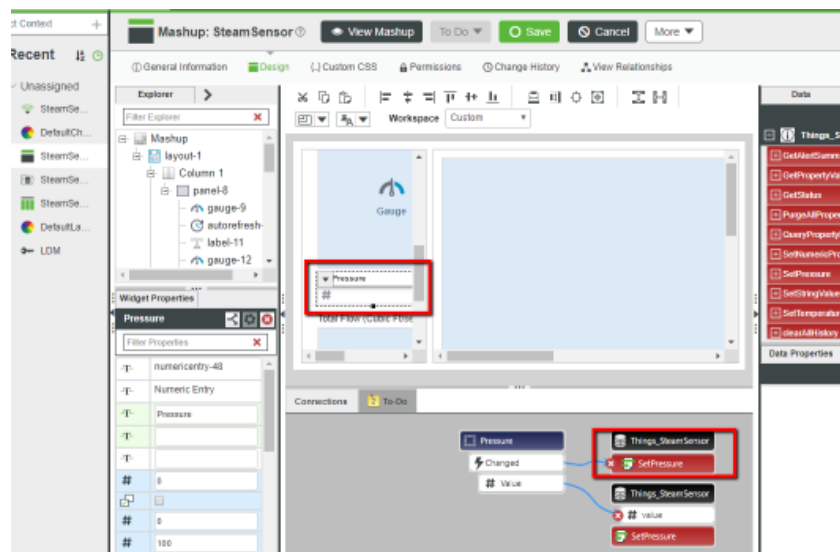

Figure: 30 Call to Local Service SetPressure

9   Firmware Details

9.1   Firmware Contents

The .firmware file downloaded to the MVI69E-LDM contains a number of files sourced from the **C:\Workspace** folder. Although they are installed on the module with the module webpage's Firmware Download process, the files can be FTP'd independently as well.

If the MVI69E-LDM device does not have any custom software already installed, then the ThingWorx-LDM sample application can be installed and started by upgrading the firmware file to the one that was built according to Chapter 4. When the module is rebooted after the upgrade, the sample application starts (by script **/etc/init.d/S88-tw**).

If the module has other custom files and/or a Firmware Update is not desirable, then the following files from **C:\Workspace** can be installed over FTP connection to the module.

#	File Location on Windows 10 PC	FTP to Folder on MVI69E-LDM	Description
1	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\Release\tw-ldm-sample-app-mvi69e	/psft/sample/tw	ThingWorx-LDM sample application
2	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\ config.json	/psft/sample/tw	Configuration file. Its structure is described in detail in the section Structure of the configuration file.
3	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\ root_ca.cer	/psft/sample/tw	Root CA certificate of the server's SSL certificate's chain.
4	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\ S10-syslog	/etc/init.d	Optional, for logging. To enable logging, service **syslog** should be started. The service can be started automatically at system reboot by copying of the script file **S10-syslog** to the folder **/etc/init.d**. Note that the script configures log file location as /www/html/log/messages.txt, i.e. under embedded web server's content location. This allows viewing of the log file's content in a web browser at URL **http://192.168.0.250/log/messages.txt** (the IP address of the module needs to be corrected accordingly).
5	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\ S88-tw	/etc/init.d	This script starts the sample application on the MVI69E-LDM after module reboot.
6	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\ subscribed_properties.bin	/opt/thingworx	Note: if the folder **/opt/thingworx** does not exist, it needs to be created.
7	C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\firmware\ mvi69e-ldm.firmware<version_	/psft/sample/tw	Note: Create the folder /psft/sample/tw on the MVI69E-LDM. Sample application executable binary, used for the Firmware Update process on the module's webpage. Generally, this file is not FTP'ed to the module.

9.2   Running of Sample Application

After installation and configuration is complete, the sample application can be started auto-matically after device reboot (if the script **/etc/init.d/S88-tw** was installed), or it can be started manually via telnet terminal, by navigating to the folder `/psft/sample/tw` and running the fol-lowing command:

`./tw-ldm-sample-app-mvi69e`

> **Note:** Power-cycling the module may cause it to lose the previously set System date and time. The "Reboot" command in the Linux OS will keep the module's previously set date and time.

10 Visual Studio 2017

Optionally, you can use Visual Studio to build the sample application. Ensure the Prerequisites and Development Environment Setup is done first.

10.1 Build Using Visual Studio

The Visual Studio 2017 solution file, located at **C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\tw-ldm-sample-app-mvi69e.sln** has 2 projects:

- tw-ldm-interface-lib
- tw-ldm-sample-app-mvi69e

In the Solution Explorer, click on **tw-ldm-sample-app-mvi69e**, right-mouse-click, and choose *Properties*.

Click **COPY TO CLIPBOARD**, as it is required to login to the server. Paste it somewhere (such as Notepad) for later reference.
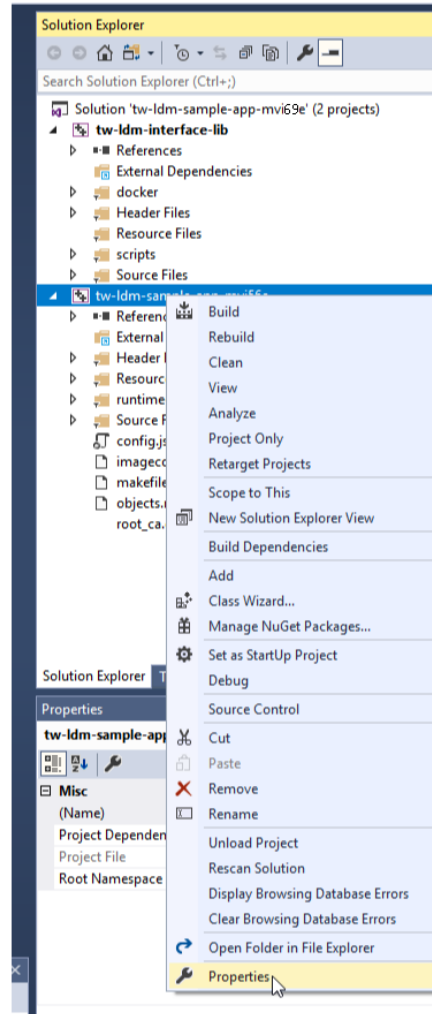
Figure: 31 Solution Explorer

1. On the left, choose Build Events. Notice the Command Line on the right. Copy the Command Line to Notepad, and modify it:

a. Set the correct IP address. This should be your PC's IP address.

b. Set the user ID.

c. Set the password.

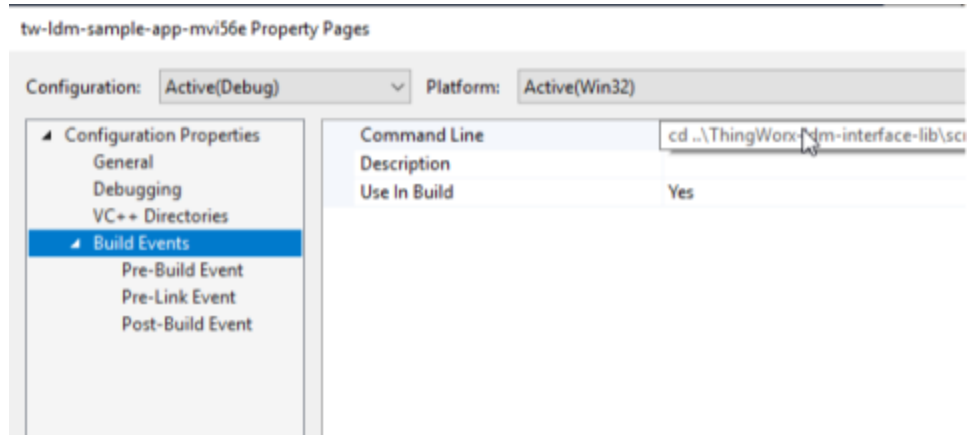2. Place the updated command line back into Visual Studio.

Figure: 32 Updated Command Back into VIsual Studio

3. Open a command prompt, choosing to Run as Administrator, and type these commands:

```
Powershell
Set-ExecutionPolicy –ExecutionPolicy RemoteSign –Scope LocalMachine
Set-ExecutionPolicy –ExecutionPolicy RemoteSign –Scope CurrentUser
```

4. In the Solution Explorer, click on tw-ldm-sample-app-mvi69e, right-mouse-click, and choose BUILD.

> Note: A warning may display, "You are building a Docker image from Windows against a non-Windows Docker host"
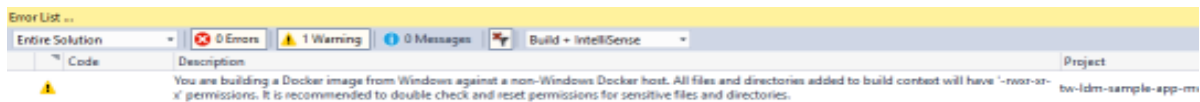
This is OK.

Figure: 33 Build

5. As a result, a Debian 9 image should be polled from Docker Hub. The required components installed to it, including toolchain, source code projects built, and firmware image are created:

C:\Workspace\ThingWorx-ldm-sample-app-mvi69e\firmware\mvi69e-ldm.firmware_<version number>_<date>.firmware

11 Service Support Warranty

11.1 Contacting Technical Support

ProSoft Technology, Inc. is committed to providing the most efficient and effective support possible. Before calling, please gather the following information to assist in expediting this process:

1. Product Version Number
2. System architecture
3. Network details

If the issue is hardware related, we will also need information regarding:

1. Module configuration and associated ladder files, if any
2. Module operation and any unusual behavior
3. Configuration/Debug status information
4. LED patterns

Details about the serial, Ethernet or Fieldbus devices interfaced to the module, if any.

Asia/Pacific	Europe/Middle East/Africa
Regional Office Phone: +60.3.2247.1898 asiapc@prosoft-technology.com Languages spoken: Bahasa, Chinese, English, Japanese, Korean REGIONAL TECH SUPPORT support.ap@prosoft-technology.com	Regional Office Phone: +33.(0)5.34.36.87.20 europe@prosoft-technology.com Languages spoken: French, English REGIONAL TECH SUPPORT support.emea@prosoft-technology.com
North Asia (China, Hong Kong) Phone: +86.21.5187.7337 china@prosoft-technology.com Languages spoken: Chinese, English REGIONAL TECH SUPPORT support.ap@prosoft-technology.com	**Middle East & Africa** Phone: +971.4.214.6911 mea@prosoft-technology.com Languages spoken: Hindi, English REGIONAL TECH SUPPORT support.emea@prosoft-technology.com
Southwest Asia (India, Pakistan) Phone: +91.98.1063.7873 india@prosoft-technology.com Languages spoken: English, Hindi, Urdu	**North Western Europe (UK, IE, IS, DK, NO, SE)** Phone: +44.(0)7415.864.902 nweurope@prosoft-technology.com Language spoken: English
Australasia (Australia, New Zealand) Phone: +60.3.7941.2888 pacific@prosoft-technology.com Language spoken: English	**Central & Eastern Europe, Finland** Phone: +48.22.250.2546 centraleurope@prosoft-technology.com Languages spoken: Polish, English
Southeast Asia (Singapore, Indonesia, Phil-	**Russia & CIS**

ippines)
Phone: +60.3.7941.2888
seasia@prosoft-technology.com
Languages spoken: English, Bahasa, Tamil

Northeast & Southeast Asia (Japan, Taiwan, Thailand, Vietnam, Malaysia)
Phone: +60.3.7941.2888 neasia@prosoft-technology.com
Languages spoken: English, Chinese, Japanese

Korea
Phone: +60.3.7941.2888 korea@prosoft-technology.com
Languages spoken: English, Korean

Phone: +7.499.704.53.46
russia@prosoft-technology.com
Language spoken: Russian, English

Austria, Germany, Switzerland
Phone: +49.(0)1511.465.4200
germany@prosoft-technology.com
Language spoken: German, English

BeNeLux, France, North Africa
Phone: +33(0)5.34.36.87.20
france@prosoft-technology.com Languages spoken: French, English

Mediterranean Countries
Phone: +39.342.8651.595
italy@prosoft-technology.com
Languages spoken: Italian, English, Spanish

Latin and South America	North America
Brazil, Argentina, Uruguay Phone: +55.11.5084.5178 brasil@prosoft-technology.com Languages spoken: Portuguese, Spanish, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com	
Mexico Phone: +52.222.264.1814 mexico@prosoft-technology.com Languages spoken: Spanish, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com	Regional Office Phone: +1.661.716.5100 info@prosoft-technology.com Languages spoken: English, Spanish REGIONAL TECH SUPPORT support@prosoft-technology.com
Andean Countries, Central America, Caribbean, Chile, Bolivia, Paraguay Phone: +507.6427.48.38 andean@prosoft-technology.com Languages spoken: Spanish, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com	

Note: For technical support calls within the United States, ProSoft's 24/7 after-hours phone support is available for urgent plant-down issues. Detailed contact information for all our worldwide locations is available on the following page.

11.2   Warranty Information

For complete details regarding ProSoft Technology's TERMS & CONDITIONS OF SALE, WARRANTY, SUPPORT, SERVICE, and RETURN MATERIAL AUTHORIZATION INSTRUCTIONS, please see the documents at:

www.prosoft-technology/legal

Documentation is subject to change without notice.