Driver Manual

FS-8704-15 XML over HTTP Driver

APPLICABILITY & EFFECTIVITY
Effective for all systems manufactured after February 2021.

Driver Revision: 1.30
Document Revision: 3.A

MSA Safety
1991 Tarob Court
Milpitas, CA 95035
Website: www.MSAsafety.com

U.S. Support Information:
+1 408 964-4443
+1 800 727-4377
Email: smc-support@msasafety.com

EMEA Support Information:
+31 33 808 0590
Email: smc-support.emea@msasafety.com

Contents

1 Description

The Ethernet HTTP XML Driver allows the FieldServer to transfer data to and from a device over Ethernet using the HTTP/XML Driver protocol. The FieldServer can emulate either a server or client.

The XML Driver is built on HTTP web technology (Port 80) and uses pages formatted in XML syntax to respond to HTTP requests or allows XML responses to be decoded and stored. Both a client and a server are supported.

The Server Side is an XML formatted response of the internal Data Array structure contained within the FieldServer, requested from a remote client device to the FieldServer URL.

The Client Side uses a HTTP GET request to a specified URL to request XML data. The driver has the ability to decode the XML response and store different Elements uniquely identified by some attribute within the element. The data of the matching Element is stored in the FieldServer Data Arrays the matching Element is stored in the FieldServer Data Arrays.

FieldServer Mode	Comments
Server	This mode is always enabled within the XML driver and is requested by "http://<ip address>/data_arrays.xml" where <ip address> corresponds to the FieldServer IP Address.
Client	Supports multiple client connections to different URLs, with associated decoding map descriptors linked to an active GET URL request.

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	32	The Driver can establish a connection to several different IP Addresses.
Server	0	The Server Side is always enabled and defined on the connection; hence the driver does not allow any server node selection.

2 Driver Scope of Supply

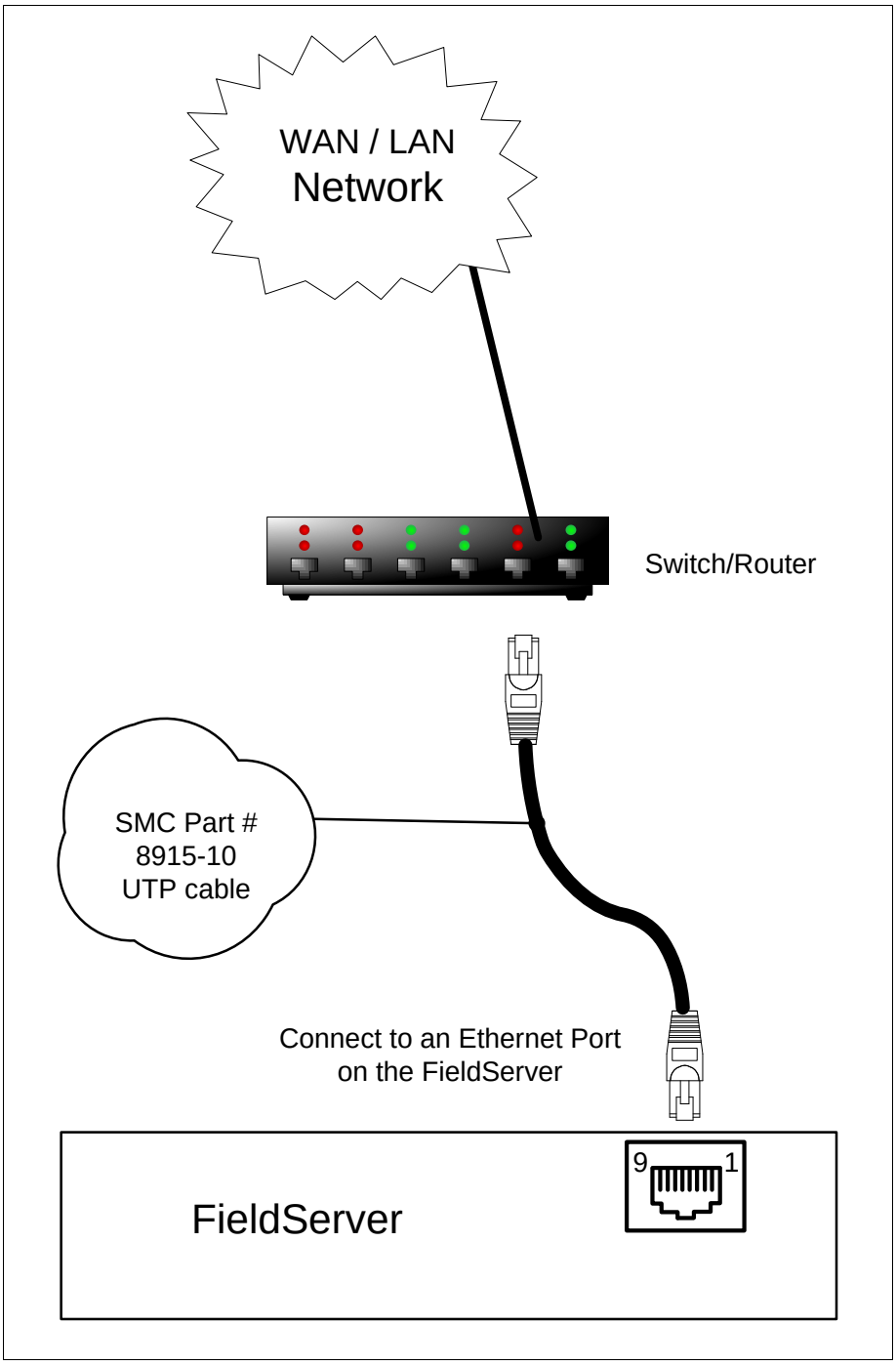
2.1 Supplied by MSA Safety

Part #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection[1]

[1] This cable is necessary for connection to the driver. It is shipped with the FieldServer and not separately with the driver.

3 Hardware Connections

The FieldServer is connected to the network as shown in connection drawing.



The Ethernet connection can be achieved using a switch or crossover cable. The driver will support all Ethernet connections on a local area network (LAN) or wide area network (WAN) including internet connections and crossover cable connections.

4 Data Array Parameters

Data Arrays are “protocol neutral” data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

Section Title		
Data_Arrays		
Column Title	**Function**	**Legal Values**
Data_Array_Name	Provide name for Data Array.	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, Byte, Uint16, Uint32, Sint16, Sint32
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10000

Example

```
//    Data Arrays
Data_Arrays
Data_Array_Name    , Data_Array_Format    , Data_Array_Length
DA_AI_01           , UInt16,              , 200
DA_AO_01           , UInt16               , 200
DA_DI_01           , Bit                  , 200
DA_DO_01           , Bit                  , 200
```

5 Client Side Configuration

For detailed information on FieldServer configuration, refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (see ".csv" sample files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an XML server running over a web server on port 80.

The configuration file tells the FieldServer about its target XML data, and the decoding of data required. In order to enable the FieldServer as an XML client, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination XML servers IP Addresses need to be declared in the "Client Side Nodes" section, and the data required from the XML requests needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

NOTE: In the following tables, * indicates an optional parameter and bold legal values are default.

5.1 Client Side Connection Parameters

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Connection/Adapter	Connection Name.	N1, N2, WLAN[2]
Protocol	Specify protocol used.	XML-HTTP, HTTP, XML

Example:

```
//    Client Side Connections
Connections
Connection              , Protocol
N1                      , XML
```

[2] Not all ports shown are necessarily supported by the hardware. Consult the appropriate instruction manual for details of the ports available on specific hardware.

5.2 Client Side Node Parameters

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for Node.	Up to 32 alphanumeric characters
IP_Address	Destination IP Address.	xxx.xxx.xxx.xxx
Host_Name*	Specify the host name of the remote device. If the Host_Name is used in place of an IP_Address, the FieldServer will attempt to resolve the IP_Address before polling remote device. If both IP_Address and Host_Name are used, the FieldServer will attempt to resolve the host name to get the latest IP_Address to use. Otherwise the configured IP_Address will be used. The FieldServer will try to resolve the host name again before starting the node recovery process to get latest IP_Address. If the host name can't be resolved, the last known IP_address is used.	Any valid host name
Response_End_Time*	Specify time duration in seconds that must elapse before a response is complete. This parameter is useful while communicating to a server that sends responses in chunks and there is a large delay (> 2s) between chunks.	1-65535; **2**
Protocol	Specify Protocol used.	XML-HTTP, HTTP, XML
Connection/Adapter	Specify which network adapter to use.	N1, N2, WLAN[3]
Remote_Node_IP_Port*	Specify port to listen to by the remote server.	0-65535; **80**
HTTP_Username*	The credentials the FieldServer must use when communicating to the HTTP server. Only basic authentication is supported.	Any alphanumeric string
HTTP_Username*	The credentials the FieldServer must use when communicating to the HTTP server. Only basic authentication is supported.	Any alphanumeric string

Example

```
//    Client Side Nodes

Nodes
Node_Name     , IP_Address      , Protocol    , Connection    , Remote_Node_IP_Port
FS_WEB_01     , 192.168.1.10    , XML         , N1            , 300
```

[3] Not all ports shown are necessarily supported by the hardware. Consult the appropriate instruction manual for details of the ports available on specific hardware.

5.3 Client Side Map Descriptor Parameters

5.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor.	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer.	One of the Data Array names from **Section 4**
Data_Array_Offset	Starting location in Data Array.	0 to (Data_Array_Length -1) as specified in **Section 4**
Function	Function of Client Map Descriptor.	Rdbc, Wrbc, Wrbx, Passive_Client

5.3.2 Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from.	One of the Node names specified in **Section 5.2**
Length	Length of Map Descriptor, Number of Elements to store into data arrays - this needs to be separated by some non-numeric character, e.g. space or comma.	Any integer.
Method*	HTTP Fetch Method for active Map descriptors.	**GET**, POST
XML-URL	HTTP URL - this does not include the IP.	Up to 200 alphanumeric characters
Write_Command*	When using POST method this is the payload of the data sent to the address.	Up to 200 alphanumeric characters (special parameters outlined in **Section 5.3.4** allowed)
Linked_Map_Descriptor	This is the active mapdescriptor responsible for fetching the XML data.	Up to 32 alphanumeric characters
Element*	XML Element to store from the XML response. The syntax is as follows "***Element.Child Element. Grand Child Element*.Attribute**" If the attribute is left out, the driver stores the Element contents. If left out, the driver stores the Data of the furthest child of the matching element.	Up to 200 alphanumeric characters
Search_Value*	String search value to uniquely match the Element the driver want to store, this could be any Attribute of the XML response. If left out, the driver uses the first element to store the data.	Up to 200 alphanumeric characters
Search_Element*	Used to specify which Element the driver searches for a match.	Up to 200 alphanumeric characters
Search_Attribute*	Used to specify which Attribute the driver searches for a match.	Up to 200 alphanumeric characters
HTTP_RESP_DA	This data array stores the HTML response from a remote device after a Write.	Up to 32 alphanumeric characters

5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled.	≥0.001s

5.3.4 Special Keywords for URL and Write_Command

Parameter	Description
<DATA_ARRAY_NAME.OFFSET>	Substitute entry with value as float from matching data array and offset.
<DATA_ARRAY_NAME.OFFSET.C>	Substitute entry with value as ASCII character from matching data array and offset. Useful for embedding special characters like < > or " etc.
<local>	Substitute value with Map descriptors first data array entry, or data offset from match cache block write.
<offset>	Used for passive map descriptors to indicate which offset changed when the cache block gets created.

5.4 Map Descriptor Examples

```
//    Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Name      , Function      , Data_Array_Name      , Data_Array_Index      , Node_Name      , Length
CMD_DA_GET1              , RDBC          , DA_HTTP_STR1         , 0                     , FS_WEB_01      , 500

, Scan_Interval      , XML-URL              , Method
, 0                  , DATA_ARRAYS.XML      , GET
```

Primary read map descriptor with URL. And method is a GET. HTTP response is stored to the data array. This is mainly for diagnostic purposes.

```
Map_Descriptors
Map_Descriptor_Name      , Function          , Data_Array_Name      , Data_Array_Index      , Node_Name
CMD_RD_UINT16            , PASSIVE_CLIENT    , DA_UINT16            , 0                     , FS_WEB_01

, Length      , Linked_Map_Descriptor
, 100         , CMD_DA_GET1
```

Linked Map Descriptors are also where the XML request is stored.

```
Map_Descriptors
Map_Descriptor_Name      , Element                          , Search_Value      , Search_Attribute
CMD_RD_UINT16            , DATA_ARRAYS.DATA_ARRAY.DATA      , DA_AI_01          , name

, Method      , XML-URL      , Write_Command
, POST        , post.cgi     , DA_UINT16.<offset>=<local>
```

XML Element to store, search value to match on, and XML attribute to match.

If the data changes, do a POST to the following URL with the payload.

```
// Write section
Map_Descriptors
Map_Descriptor_Name   , Function      , Data_Array_Name      , Data_Array_Index      , Node_Name      , Length      , Scan_Interval
CMD_WR_UINT16a        , WRBC          , DA_UINT16_WR         , 0                     , FS_WEB_01      , 10          , 2s

...                   , Method      , XML-URL      , Write_Command
...                   , POST        , post.cgi     , "DA_UINT16.0=<DA_UINT16_WR.0>&DA_UINT16.1=<DA_UINT16_WR.1>"
```

URL and payload of a regular POST command generated by the FieldServer.

6 Configuring the FieldServer as a Server

For detailed information on FieldServer configuration, refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (see ".csv" sample files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an XML client. This can be any web browser such as Internet Explorer or Firefox. Or this can be any other XML client application. The server will listen on the same port (80/443) and protocol (HTTP/HTTPS) as the web server for the device. This can be set up on the FS-GUI page, under the security tab.

NOTE: In the tables below, * indicates an optional parameter with the bold legal value as default.

6.1 Server Side Connection Parameters

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Connection/Adapter	Connection Name.	N1-N2, WLAN[4]
Protocol	Specify protocol used.	**XML-HTTP**, HTTP, XML
Max_Sessions*	Specify the maximum number of sessions.	Any integer value; **100**

Example

```
//   FieldServer Driver specific parameters

Bridge
Connection                              , Protocol
N1                                      , XML
```

6.2 Server Side Node Parameters

Nodes do not apply to the server side of this protocol.

6.3 Server Side Map Descriptor Parameters

Map descriptors do not apply to the server side of this protocol.

[4] Not all ports shown are necessarily supported by the hardware. Consult the appropriate instruction manual for details of the ports available on specific hardware.

6.4 Server Side Requests and Responses Supported

Request:

```
http://192.168.1.5/data_arrays.xml
```

Response:


```
<DATA_ARRAYS FST_XML_VERSION="1.00" MAX_INDEX="5" BRIDGE_TITLE="Lonworks Server">
    <DATA_ARRAY NAME="DA_Group" FORMAT="Byte" LENGTH="99" INDEX="1">
        <DATA OFFSET="0" DATA_AGE="3:00:39:36.736s" STATUS="0">0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</DATA>
    </DATA_ARRAY>
    <DATA_ARRAY NAME="DA_P16_1Sta" FORMAT="Byte" LENGTH="2" INDEX="2">
        <DATA OFFSET="0" DATA_AGE="34.705s" STATUS="0">0 0</DATA>
    </DATA_ARRAY>
    <DATA_ARRAY NAME="DA_Pnl16_1" FORMAT="Byte" LENGTH="8" INDEX="3">
        <DATA OFFSET="0" DATA_AGE="3:00:39:36.646s" STATUS="0">0 0 0 1 0 0 0 0</DATA>
    </DATA_ARRAY>
    <DATA_ARRAY NAME="DA_P16_2Sta" FORMAT="Byte" LENGTH="2" INDEX="4">
        <DATA OFFSET="0" DATA_AGE="34.050s" STATUS="0">0 0</DATA>
    </DATA_ARRAY>
    <DATA_ARRAY NAME="DA_Pnl16_2" FORMAT="Byte" LENGTH="8" INDEX="5">
        <DATA OFFSET="0" DATA_AGE="3:00:39:36.548s" STATUS="0">1 0 0 0 0 0 0 0</DATA>
    </DATA_ARRAY>
</DATA_ARRAYS>
```


Section Title		
Optional Selective Data_Array.xml parameters		
Parameter	**Function**	**Legal Values**
?name=DA_NAME	Specify which Data array to retrieve.	Up to 15 alphanumeric characters.
:LENGTH	Number of values to return.	**>=1**
.OFFSET	Start at which offset in the Data Array.	**>=0**
&name= …	Add another Data Array to the response. When adding additional Data Arrays, Length and Offset parameters must also be included.	Up to 15 alphanumeric characters.
STATUS	Validity status of data. See **Section 8** for BITS table.	2 = Invalid 272 = Valid, Waiting 400 = Valid, Waiting, Offline 912 = Valid, Waiting, Offline and Old

<u>Example</u>

```
http://192.168.1.5/data_arrays.xml?name=DA_REG01.01:5&name=DA_REG02.01:5
```

7 XML Server Schema

Below is the XML schema used for server responses to the data_Arrays.xml request.

```
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="DATA_ARRYS">
   <xs:attribute name="FST_XML_VERSION" type="xs:string" />
   <xs:attribute name="MAX_INDEX" type="xs:string" />
   <xs:attribute name="BRIDGE_TITLE" type="xs:string" />
   <xs:complexType>
     <xs:sequence>
       <xs:element name="DATA_ARAY">
         <xs:attribute name="NAME" type="xs:string" />
         <xs:attribute name="FORMAT" type="xs:string" />
         <xs:attribute name="LENGTH" type="xs:string" />
         <xs:attribute name="INDEX" type="xs:string" />
         <xs:complexType>
           <xs:element name="Data" type="xs:string">
             <xs:attribute name="OFFSET" type="xs:string" />
             <xs:attribute name="DATA_AGE" type="xs:string" />
             <xs:attribute name="STATUS" type="xs:string" />
           </xs:element>
         </xs:complexType>
       </xs:element>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

8 Status Parameter BITS Table

Data Status	Hex Value	Description
DATA_BAD	0x0002	Data considered invalid due to one or multiple reasons. For example: a bad sensor, out of range analog input, trouble, reliability, etc.
DATA_VALID	0x0010	Data valid
DATA_COPIED	0x0020	Copy of data made
DATA_FNC_AFTER_STORE	0x0040	Function after store required on this data array.
DATA_OFFLINE	0x0080	Offline
DATA_WAITING	0x0100	Waiting
DATA_OLD	0x0200	Old
DATA_EXPIRED	0x0400	Expired
DATA_STARTUP	0x0800	Startup
DATA_UNRELIABLE	0x1000	Data Reliability

9 Setting the Format of the Data Array Age

By default the data age of a data array starts at 'S.SSS' and grows to 'M:SS.SSS', 'H:MM:SS.SSS' or 'D:HH:MM:SS.SSS' accordingly. The Disp_Time_Format defines if the time format follows H:MM:SS.SSS or D:HH:MM:SS.SSS.

Section Title		
Bridge		
Parameter	**Function**	**Legal Values**
Title	Allows user to specify the title of the FieldServer.	Title Text
Disp_Time_Format	Time format of the data array age.	HH:MM:SS.SSS, **D:HH:MM:SS.SSS**

Example

```
//=============================================================================
//
//    Common Information
//

Bridge
Title             , Disp_Time_Format
HTTP WEB/XML Server , HH:MM:SS.SSS
```

10 Use of SSI/TLS for Secture Connection

SSL/TLS (Secure Sockets Layer/Transport Layer Security) is a security technology for establishing an encrypted connection between a server and a client. This allows the secure transfer of data across untrusted networks.

These functions are supported on the following:

FS-QS-1010 or **FS-QS-1210** with a serial number starting with 14 or later (indicating the year it shipped).

FS-QS-1011 or **FS-QS-1211** with a serial number starting with 15 or later (indicating the year it shipped).

Minimum BIOS requirement: 2.6.1

10.1 Configuring FieldServer as a SSL/TLS Server

The following example sets the FieldServer to accept a secure Modbus/TCP connection on port 1502.

10.1.1 Simple Secure Server Configuration

Add TLS_Port parameter in the connections section of the configuration file and set to a port number between 1 – 65535.

```
Connections
Adapter    , Protocol          , TLS_Port
N1         , Modbus/TCP        , 1502
```

This configuration sets the FieldServer to accept any incoming connection but will not request a client's certificate for verification. This means that the FieldServer end point communication will be encrypted but not authenticated.

The FieldServer will send an embedded self-signed certificate if one is requested by a connecting client.

NOTE: If a remote client requires a certificate, then request the smc_cert.pem certificate from MSA Safety Technical Support and update the remote client's authority as per vendor instructions.

10.1.2 Limiting Client Access

In addition to TLS_Port parameter also add Validate_Client_Cert in the connections section of the configuration file and set it to “Yes”.

```
Connections
Adapter    , Protocol          , TLS_Port      , Validate_Client_Cert
N1         , Modbus/TCP        , 1502          , Yes
```

The configuration above sets the FieldServer to request and verify a client’s certificate against its internal authority file before accepting connection. By default, this means the FieldServer will only accept connections from other FieldServers.

In order to load an authority file so that the FieldServer will accept connections from a chosen list of remote clients, configure the FieldServer with the following connection settings:

```
Connections
Adapter    , Protocol          , TLS_Port    , Validate_Client_Cert      , Cert_Authority_File
N1         , Modbus/TCP        , 1502        , Yes                       , my_authorized_clients.pem
```

This configuration has the FieldServer accept connections from clients who have the correct certificate. The authority file is a collection of client certificates in PEM format. This file can be edited using any text file editor.

NOTE: Cert_Authority_File is useful only if Validate_Client_Cert is set to ‘Yes’.

10.1.3 To Upload the Authority File to the FieldServer

1. Enter the IP address of the FieldServer into a web browser.
2. Choose the ‘Setup’ option in the Navigation Tree and Select ‘File Transfer’.
3. Choose the ‘General’ tab.
4. Click on the ‘Browse’ button and select the PEM file you want to upload.
5. Click on ‘Submit’.
6. When the message, “The file was uploaded successfully” appears, click on the ‘System Restart’ button.

10.1.4 Certificate Validation Options

If connections must be limited to only a particular domain (vendor devices), include Check_Remote_Host to specify the domain/host name.

```
Connections
Adapter , Protocol      , TLS_Port , Validate_Client_Cert , Cert_Authority_File          , Check_Remote_Host
N1      , Modbus/TCP    , 1502     , Yes                 , my_authorized_clients.pem , SMC
```

The configuration above tells the FieldServer to only accept connections that have the correct certification and is coming from the specified host.

The Check_Remote_Host value is synonymously known as common name, host name or domain etc. The common name can be obtained by the following methods:

- Ask the certificate issuer for the host name.
- Use online tools to decode the certificate (for example: https://www.sslshopper.com/certificate-decoder.html).
- If the program openssl is installed on the local PC, then run the following command to get the common name: openssl x509 -in certificate.pem -text -noout

10.1.5 Certificate Validation Options

Make sure the certificate is in PEM format. Otherwise, convert it to PEM format (reference the link below). support.ssl.com/Knowledgebase/Article

Configure the FieldServer to use a custom certificate as shown below:

```
Connections
Adapter , Protocol      , TLS_Port , Server_Cert_File
N1      , Modbus/TCP    , 1502     , my_server_cert.pem
```

10.2 Configuring FieldServer as SSL/TLS Client

The following Node configurations set the FieldServer to open a secure Modbus/TCP connection to the server at IP Address 10.11.12.13 on port 1502.

10.2.1 Simple Secure Client Configuration

Add Remote_Node_TLS_Port parameter in the nodes section of the configuration file and set to a port number between 1 – 65535.

```
Nodes
Node_Name   , Node_ID   , Protocol     , Adapter   , IP_Address    , Remote_Node_TLS_Port
PLC_11      , 11        , Modbus/TCP   , N1        , 10.11.12.13   , 1502
```

The above configuration sets the FieldServer to connect to a remote server but does not request a server's certificate for verification. This means that the FieldServer end point communication will be encrypted but not authenticated.

If requested by a remote server, the FieldServer will send an embedded self-signed certificate.

10.2.2 Limit Server Access

Add the Validate_Server_Cert parameter to the client node section of the configuration.

```
…….   , Remote_Node_TLS_Port      , Validate_Server_Cert
……..  , 1502                     , Yes
```

The above configuration sets the FieldServer to request and verify the server's certificate against its own internal authority file before finalizing the connection. By default, this means the FieldServer will only establish connections to other FieldServers.

```
…….   , Remote_Node_TLS_Port      , Validate_Server_Cert      , Cert_Authority_File
……..  , 1502                     , Yes                      , my_authorized_servers.pem
```

The above configuration sets the FieldServer to use a specified PEM file to allow custom server connections.

The authority file is a collection of server certificates in PEM format. This file can be edited using any text file editor (such as notepad). When the file has all required certificates, paste it into the PEM formatted server certificate. Now the FieldServer will connect to a server if it can find the server's certificate in the authority file.

NOTE: Cert_Authority_File is useful only if Validate_Client_Cert is set to 'Yes'.

To upload the Certificate to the FieldServer follow the directions for the authority file in **Section 10.1.3**.

10.2.3 Certificate Validation Options

Use the Check_Remote_Host element as described in **Section 10.1.4**.

10.2.4 Set up Client Certificate

Make sure the certificate is in PEM format. Otherwise, convert it to PEM format (reference the link below).
support.ssl.com/Knowledgebase/Article

Configure the FieldServer to use a custom certificate as shown below:

```
……..   , Client_Cert_File
……..   , my_client_cert.pem
```