A Sierra Monitor Company

Driver Manual

(Supplement to the FieldServer Instruction Manual)

FS-8700-109 PCP Complete Protocol

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after April 2010

Driver Version:	1.00
Document Revision:	9

TABLE OF CONTENTS

1 PROTOCESSOR COMPLETE PROTOCOL (PCP) DESCRIPTION

The PCP Serial Protocol is designed to achieve the following in a very simple manner:

- Read a Customer Device's Database of data objects (Discovery)
- Auto create a Client configuration to read the values/states of the data objects discovered
- Allow data transfer (read and write) between the Driver and the customer Device
- Auto create a Server configuration to serve the data to other Client devices using protocols e.g.Metasys, Lonworks or BACnet.
- Determine if the database has changed

The driver can be configured manually to read/write data.

The driver can also be used in a simple mode to poll for specific data from a customer device. To achieve this, the data objects in the customer device must be known in advance.

The driver provides both Client and Server emulation. The Server side of the driver is intended to support FieldServer's Quality Assurance program and is not intended to provide complete emulation of a PCP Server. Thus the Server side is not fully documented. At a customer's request the Server side functionality can be documented and enhanced.

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	1	The protocol is node-less. This means that messages do not contain source/destination node addresses and the driver cannot differentiate between messages from multiple devices on the same connection. Thus only one node may be defined per connection.
Server	1	

1.1 Getting started:

1.1.1 Review the protocol specification

The protocol specification document is available from FST. Please request the document named: PS002 - PCP Driver Spec (FS-8700-109). Read and review the document.

1.1.2 Program the Device to respond to messages sent by the Driver

The device must be programmed to respond to the messages sent by the driver. This is done using the serial port API of the device.

Poll From ProtoCessor	
11	Read Property Values
12	Read All Property Values
21	Write Property Values

The Driver can "Discover" an existing database or can be configured to read data a fixed set of data objects (points). In order to support "Discovery", the device must be programmed to respond to the following queries sent by the Driver.

Poll From ProtoCessor	
01	Poll Object Class Count
02	Poll Object Class Configuration
03	Poll Object Class Property
04	Poll Object Instance Count
05	Poll Object Instance Info
06	Configuration Valid / Changed

1.1.3 Program the device's serial port to send responses

1.1.4 Configure the driver to do discovery and to read data from your device.

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by FieldServer Technologies for this driver

FieldServer Technologies PART #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection

3 HARDWARE CONNECTIONS

The ProtoCessor is integrated into the customer's hardware at design time. No additional connection information is required.

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, UInt16, SInt16, Byte.
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.	1-10, 000

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 CONFIGURING THE FIELDSERVER AS A PCP CLIENT

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

This section documents and describes the parameters necessary for configuring the ProtoCessor to communicate with a PCP protocol enabled device.

5.1 Client Side Connection Parameters

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[1]
Protocol	Specify protocol used	PCP, PCP_Adv, Pcp_Advanced
Baud*	Specify baud rate	110; 300; 600; 1200; 2400; 4800; 9600; 19200; 28800; **38400**; 57600; 115200
Parity*	Specify parity	Odd, **Even**, None
Data_Bits*	Specify data bits	7,**8**
Stop_Bits*	Specify stop bits	**1**,2
Poll _Delay*	Time between internal polls	0-32000 s, **1 s**
Auto_Config_Client*	This parameter instructs the driver to auto create Data Arrays and Map Descriptors to read the values/states of the objects discovered in the Server node. Refer to Appendix A.1	Yes, Fast, **No**
Auto_Config_Server*	Used in conjunction with Auto_Config_Client, the driver auto creates Server Side Connections, Nodes and Map Descriptors to serve the data read by the auto configured client. Refer to Appendix A.1.2	BACnet-IP, BACnet-Ethernet, BACnet-MSTP, MN2, **No**
Extra_Timeout_Control*	To reduce the number of errors displayed set this parameter to 1. This causes the driver to re-poll once per timeout before the timeout becomes an official error. See Appendix A.4	**No**, 1
Server_Object_ID_Style*	This parameter allows the user to determine the method form numbering Server Side objects. Additional notes are provided in Appendix A.1.3	**0**,1,2,3

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

Example

Suitable for polling a Server Node where the database is known in advance.

```
//        Client Side Connections

Connections
Port    , Protocol    , Baud      , Parity
P1      , PCP         , 38400     , None
```

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
Node_ID	This parameter is used when a Server-Side is auto-created.	Refer to Appendix A.1.2
Protocol	Specify protocol used	PCP, PCP_Adv, Pcp_Advanced
Connection	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[2]

Example

```
//   Client Side Nodes

Nodes
Node_Name    , Protocol    , Connection
Controller1  , PCP         , P1
```

5.3 Client Side Map Descriptors

5.3.1 FieldServer Related 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 "Data Array" section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor.	Use read commands e.g. Rdbc, for commands: 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x11, 0x12, Use write commands e.g. Wrbc, Wrbx for commands: 0x21

[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.

5.3.2 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	Each Map Descriptor defines storage locations for a series of addresses. This parameter specifies the length of the series.	1,2,3 .etc Whole numbers
PCP_Data_Type	When data is transferred between the PCP server and client, the value or status for each property is transferred using a 2 byte field. This parameter tells the driver how to interpret this field. Used to define the Server object type when the driver auto creates a Server. Tells the driver how to unpack the field when responding to polls. Tells the driver how to prepare the two byte field when sending data. More information is provided in Appendix A.3	BIT, BYTE, UINT, SINT, F.1, F.2 , F.3
PCP_Cmd	Specifies the command that must be executed by the Map Descriptor. The protocol specification document provides additional details. A Command Summary is provided in section Appendix C A special command is used by the driver to read the database (0x00). This command is not defined in the protocol spec but rather instructs the driver to proceed through a sequence of commands 0x01...0x06 to read the full database from the PCP Server.	Can be specified in Hex e.g. 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x11, 0x12, 0x21 Can be Specified in Decimal 1, 2, 3, 4, 5, 6, 17, 18, 33
Class_Type*	The meaning of this parameter is context sensitive. For commands 0x01 to 0x06 the parameter defines an Index number e.g. The 0'th Class, the 1st Class, etc For commands 0x11,0x12, 0x21 the parameter defines the Class Number e.g. Class whose number is 1, class whose number is 2.	Whole Numbers
Inst_Num*	The class instance number.	Whole Numbers
Prop_Num*	The property number being read/written. Only required for commands 0x11, 0x12, 0x21	Whole Number
Prop_Index*	Used by command 0x05. An index number e.g. the 0th property, the 1st property	Whole Numbers
AutoCreated*	When the driver auto-creates client side Map Descriptors it marks them. This is intended as a trouble shooting tool and should not be included in the configuration file.	Yes, **No**

5.3.3 Timing Parameters

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

5.3.4 Map Descriptor Example 1 – Read Database

The PCP_Cmd = 0x00 tells the driver to read the PCP Server's database. The driver uses commands 0x01.0x06 to achieve this. The driver dumps the database in ASCII in the specified Data Array. View the array in 'String format.

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Length   , PCP_Cmd
Read_Database         , DA_PCP_DUMP       , 0                   , Rdbc       , CPU1        , 2000     , 0x00
```

If the database requires more space it will be truncated. Set the length to 1 if not using the dump.

This command tells the driver to read the database.

5.3.5 Map Descriptor Example 2 – Read Data

This Map Descriptor will read a list of Classes, Instances and Properties directly. In this example, data from the 1st instance of class #7's property #24 is read. Specify the correct data type or the 2 byte value field in the response will be decoded incorrectly. The Scan_Interval is not specified so the default is assumed.

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Length   , PCP_Cmd   , Class_Type   , Inst_Num   , Prop_Num   , PCP_Data_Type
Read_C7I1P24          , DA_BUIDING01      , 109                 , Rdbc       ,             , 1        , 0x11      , 7            , 1          , 24          , UINT
```

Data is stored in this data array at location 109.

If the length was 2 then property 25 would be read and stored at offset 110. **Ensure that property 25 has the same data type**

Read Property Command.
To Write use 0x12 and change the function to wrbc.

1st instance of class #7's property #24

5.3.6 Map Descriptor Example 3 – Read All Class Data

Use of this command is not recommended unless all the properties of the class being read have the same data type. The Map Descriptor reads data from the 1st instance of class #7. The properties are stored sequentially starting at the offset specified. Ensure the length is sufficient – if there are more items in the response than in the specified length, data items will be discarded.

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Length   , PCP_Cmd   , Class_Type   , Inst_Num   , PCP_Data_Type
Read_C7I1P24          , DA_BUIDING01       , 0                   , Rdbc       , CPU1         , 100      , 0x12      , 7            , 1          , UINT
```

Ensure length is sufficient to list all the properties in the response.

Bit - Decodes as – If (non-zero) store 1 else store 0
Byte - Decodes as – Store lsb
UINT -
SINT -
F.1 - Decodes as – store (word interpreted as uint16 value * 10.0F)
F.2 - Decodes as – store (word interpreted as uint16 value * 100.0F)
F.3 - Decodes as – store (word interpreted as uint16 value * 1000.0F)

Appendix A. Useful Features

Appendix A.1. Auto Create Client and Server

Appendix A.1.1. Auto Create Client

If the ProtoCessor is configured to read the PCP server's database it can auto-create Client side Map Descriptors and Data Arrays. The auto-creation occurs when the database read is complete. The Client side Map Descriptor's can be created in one of two styles depending on the value of the connection parameter 'Auto_Config_Client'.

Parameter Value	Style
Yes	The driver creates one MAP DESCRIPTOR for each property of each class instance using command 0x11 (Read Property Value).
Fast	The driver creates one MAP DESCRIPTOR for each class instance using command 0x12 (Read All Property Values).

When the client is created the driver creates a file called auto.txt which can be uploaded from the ProtoCessor. The file contains a listing of all the Map Descriptor's created.

Naming conventions:

- Data Arrays: DA_Cxx_Iyyy where xx is the class number and yyy is the instance number.
- Map Descriptors Xxxxxxx[yyy] where xxxxxx is the class name and yyy is the instance number.

Appendix A.1.2. Auto Create Server

A driver configured to auto-create a client, can auto-create the BACnet Server side as any of BACnet-IP, BACnet-Ethernet, BACNet-MSTP and Metasys N2 Open (MN2) Server. All Map Descriptors created are listed in the auto.txt file which can be uploaded from the ProtoCessor.

The auto-creation process does the following

- Creates a connection using adapter N1
- Creates a Node using the Node_ID of the PCP node.
- Creates one Map Descriptor for each data object.
    - The object type is based on the property type - See Appendix A.3
    - The name of the Map Descriptor and hence the name of the object is formed as follows:

S_xxxxxxx[yyyyy].zzzzzzzzzz where
S_ is a prefix added to all these Map Descriptor's
xxxxxxxxx is the name of the class
Yyy is the instance number
Zzzzzzzzz is the name of the property
e.g.: MuaII[048].FanInStartUpDelay

Appendix A.1.3. Configuration Example

In conjunction with a polling Map Descriptor, the driver will read the database, create Data Arrays and Map Descriptors to poll for values/states for each object/property discovered and create Server Side Connection, Node and Map Descriptors to serve this data to remote BACnet clients.

```
//      Client Side Connections

Connections
Port   , Protocol   , Baud    , Parity   , Auto_Config_Client   , Auto_Config_Server
P1     , PCP        , 38400   , None     , Yes                  , BACnet-IP
```

Appendix A.2. Server Object ID Style

When the Driver auto creates the Server objects for BACnet it has to number them. The numbering of these objects can be controlled by specifying the 'Server_Object_ID_Style' parameter on the connection. The highest object ID that a BACnet object may have is approximately 4,000,000. Ensure that an option is selected that does not produce object ID's larger than this limit.

Style	Numbering
0 (Default)	The driver numbers objects of each data type sequentially starting at 1. Example AI1, AI2, AI3, AO1, AO2, AO3…etc. If the server protocol is Metasys N2 Open (MN2) then this is the only permitted numbering style.
1	The driver uses PCP Class, instance and property number to form the object number using the following formula: Object_ID = Class_Number * 100000+ Instance_Number * 1000 + Property_Number E.g: 7011001 for class number 7 instance 11 property 1. The object number is not dependent on class.
2	The driver uses PCP Class, instance and property number to form the object number using the following formula: Object_ID = Class_Number * 20000+ Instance_Number * 200 + Property_Number E.g. 7011001 for class number 7 instance 11 property 1. The object number is not dependent on class.
3	The driver uses PCP Instance and property number to form the object number using the following formula: Object_ID = Instance_Number * 1000 + Property_Number E.g.7011001 for class number 7 instance 11 property 1. The object number is not dependent on class. This option is suitable for WattMaster devices.

Appendix A.3. Data/Object Type

The Data Type plays two roles

- Tells the driver how to prepare/unpack the two byte data value field in messages which transfer data.

- Is used by the driver when Server side data objects are auto-created by the driver.

Appendix A.3.1. Data Type Selection

When data is transferred between the PCP Server and Client, the value or status of each property is always transferred using a 2 byte field. The PCP_Data_Type parameter tells the driver how to interpret this field. In the case of response to polls, this parameter tells the driver how to unpack the field. When the driver sends data, this parameter tells the driver how to prepare the two byte field. **It is easy to see that in a situation where the client and server are using different data types that data will be misinterpreted**.

Where the driver is configured to read the database, the driver will discover the class, instances and properties in the database. This information includes the data type of each property discovered. The driver will use this data type, if possible, and thus it isn't necessary to know the data type in advance. In cases where the driver does not read the database, it is necessary to obtain the data type of each from the vendor.

Appendix A.3.2. Use of Data Type in creating Server Side Objects.

When BACnet Server side objects are created they are allocated a data type according to the following rules. If the Class Type is less than 6 then all properties inherit the type of the class.

Class Type Number	Class Type	Class Property Data Types
1	AI	AI
2	AO	AO
3	AV	AV
4	BI	BI
5	BO	BO
6	BV	BV
7, 8 etc …	User Defined.	Derived from the property type reported by the PCP Server when the database is read. If the Property Data Type is 'Bit' then the object is created as a Binary Object – BI if read only otherwise a BV. All other data types give rise to Analog Objects - AI if read only otherwise AV.

The PCP server indicates that the data type of a property is read/write by setting the high nibble of the data type to 1.

Appendix A.4. Extra Timeout Control

When a client poll times out, a timeout is produced which presents as an error. Some customer CPU's are not fast enough to handle every incoming poll correctly. To reduce the number of errors displayed set this parameter to 1. This causes the driver to re-poll once per timeout before the timeout becomes an official error.

Official timeouts contribute to the node status. If there are x official timeouts then the node is considered offline. If there is even 1 official timeout then the driver enters the retry interval which by default is slower than the normal polling rate

Appendix B. Troubleshooting

- Servers typically respond with NAK's when they receive badly formatted polls.
- Servers typically respond with NO_DATA responses when polled for data that doesn't exist.
- Servers typically don't respond at all when the message they receive is so badly formatted that they don't know they have received a full message. CPU's with marginal speed sometimes do not receive complete messages and therefore don't respond at all. For this reason a special method of handling a timeout has been developed for this driver. See Appendix A.4 for more information.
- If the driver has been configured to read a database then it will not process other Map Descriptor's until the database read is complete.
- To watch the driver storing data each time a response is received, activate stat#20. (See Appendix B.2)

Appendix B.1. Driver Error Messages

Some configuration errors produce an error every time a poll is generated. This would fill the error buffer quickly without adding clarity. The driver therefore suppresses subsequent similar messages so that the same error produced by multiple Map Descriptors produces only one error message. Subsequent error messages can be seen on the driver message screen.

Note : In the actual message you will see that %d has been replaced by an integer, %s by text indicating a Data Array or Map Descriptor name and %x by two hex characters.

Error Message	Description and Action Required
PCP: #01 Err. Bad MD Len - defaulting to 1	Map Descriptor length is zero or not set. Driver assumes length as 1.[3]
PCP: #01a Err. Illegal Map Descriptor length, Max 30	If Map Descriptor is defined with PCP_Cmd as 0x21 (Set), maximum length is restricted to 30.[3]
PCP: #01b Err. Illegal Map Descriptor length Max 40	If Map Descriptor is defined with PCP_Cmd as 0x11 (Get), maximum length is restricted to 40.[3]
PCP: #02 Err. Undefined PCP_Cmd	PCP_Cmd parameter is blank in configuration file.[3].
PCP: #02a Err. PCP_CMD badly formatted	PCP_Cmd parameter's value is badly formatted. Note: hex format is 0x?? or 0X??.[3]
PCP: #03 Err. Undefined Class_Type	Class_Type parameter is blank in configuration file.[3]
PCP: #04 Err. Undefined Inst_Num	Inst_Num parameter is blank in configuration file.[3]
PCP: #05 Err. Undefined Prop_Num	Prop_Num parameter is blank in configuration file.[3]
PCP: #06 Err. Undefined Prop_Index	Prop_Index parameter is blank in configuration file.[3].

[3] Edit the .CSV file, download the modified file and reset the FieldServer to have the changes take effect.

Error Message	Description and Action Required
PCP: #07 Err. Undefined PCP_Data_Type	Pcp_Data_Type parameter is blank in configuration file.[3]
PCP: #08 FYI. Database Md %s	The name of the Map Descriptor responsible for getting the database from the Server.[4]
PCP: #11 Err. One BACnet Node required.	When configured to auto create a BACNet server a BACNet node is required in the configuration file.[3]
PCP: #12 Err. Heading not equal to keywords	Call Tech Support.
PCP: #13 Err. MD Name too long <%s>	Map Descriptor indicated cannot be created. Call Tech Support.
PCP: #14 FYI. Re-polling on timeout mode active.	Client re-polls if it gets timeout on first poll. Requires connection is configuration.
PCP: #15 Err. Parse Failed	Client got response in unknown format. Call Tech support with other information displayed with this message.
PCP: #16 Err. Proto Err. Seq Number	Message sequence number for poll-response mis-matched.[4]
PCP: #17 Err. Diagnostic 1	Client set to poll server with first half of the request only.
PCP: #18 Err. Diagnostic 2	Client set to poll server with last half of the request only.
PCP: #19 Err. Diagnostic 3	Client set to do a poll proceeded with garbage characters.
PCP: #21a Err. Proto Err. Bad CMD code. Poll=%x Resp=%x	Command did not match while polling for database.
PCP: #21b Err. Proto Err. Bad CMD code. Poll=%x Resp=%x	Command did not match while polling for data other than database.
PCP: #21c Err. Proto Err. Bad CMD code. Poll=%x	Client made poll with unknown command.
PCP: #21d FYI. Unknown CMD %d (%2X)	Client trying to poll with unknown command.
PCP: #22 Err. Cmd0x5 Null Ptr.	Take a log and call Tech Support
PCP: #23 Err. Device reports none of polled properties exist. MD = %s	Server device retuned none of the requested properties.
PCP: #24 FYI. Confg DataBase Complete	Client received the entire configuration database.
PCP: #25 FYI. Unknown Confg Change Code %d ,assuming 1	Returned code for configuration change is non zero but not 1 - driver considers it as 1 and will proceed to get new database.
PCP: #26 Err. Cant store at %u da %s dalen %u	There is insufficient space to store in indicated Data Array.
PCP: #27a FYI. MD's <%s> bad Write Status	Response shows at least one of the properties is not written successfully.

[4] This message will be printed on Driver message screen.

Error Message	Description and Action Required
PCP: #27b FYI. %d th Property written status is 1	Indicates very first property which did not writte successfully.[5]
PCP: #28 Err. Unknown Md CMD %d	Map Descriptor is asked to store data with unknown command.
PCP: #31 Err. Asked to calc chk. 1st byte Exist/Reqd %2X/%2X	Found illegal start byte while calculating checksum.[5]
PCP: #32 FYI. You could use an Array called <%s> to expose diagnostic info	Declare indicated Data Array in Data_Array section for driver to expose diagnostic statistics or to force driver to follow diagnostic commands. See Appendix B.2
PCP: #33 FYI. You could use an Array called <%s> to dump database info	Declare indicated Data Array in Data_Array section if you want driver to dump database info in Data Array.
PCP: #34 FYI. Waiting for permission to dump DB.	Driver is waiting for permission to dump database info in a file. Set offset PCP _OK_TO_DUMP to 1 in the stats array.
PCP: #34a ERR: Cannot create dump file %s	Internal problem to create a file.
PCP: #34b ERR: Cannot open file %s	Internal problem to open a file.
PCP: #35 FYI. Dump Da insufficient space	Insufficient space in Data Array declared to store database information. Information will be truncated.
PCP: #36a FYI. Unknown class type. Treat as 'AV'	
PCP: 36b FYI. Unknown class type. Treat as 'AV'	
PCP: #36c FYI. Unknown class type. Treat as 'AV'	
PCP: #37 Err. No Memory at FieldServer	Database is too big to be handled by FieldServer. Call Tech Support.
PCP: #41 FYI. Response inhibited.	Offset PCP _STAT_RESPONSE_INHIBITED has been set to 1 in the stats array. Change to zero to remove this restriction.
PCP: #42 Err Diagnostic 4	Server set to respond with incorrect checksum.
PCP:#43 FYI Server_Object_ID_Style overridden to 'Default'	You may ignore this error if you are satisfied that default style is appropriate. Read more in Appendix A.1.3 To eliminate this error, edit the config and change the value of the parameter to 'Default' or '0'.[5].
PCP: #46 Err. Class has %d properties But is serving info for property #%d"	When the server reported its class information it reported the property count as x. Now, when polled for property attributes (using cmd=0x03), it is reporting the attributes of a property whose index is greater than x. If you are using the FieldServer as a server then report this to tech support. If you are using a Wattmaster or other Vendor device then report this error to the vendor.

[5] Edit the .CSV file, download the modified file and reset the FieldServer to have the changes take effect.

Appendix B.2. Driver stats

In addition to the standard FieldServer operating statistics the driver exposes certain key stats in a Data Array if required. A Server device can then monitor these stats. The following must be added to the configuration file to activate these stats.

```
// Expose Driver Operating Stats.

Data_Arrays
Data_Array_Name    , Data_Format    , Data_Array_Length
pcp-stats          , UINT32         , 200
```

Length must be sufficient to store error messages.

Table shows offset (= Stat Number) where stat will be stored in "pcp-stats" Data Array.

Stat Number	Stats	Description
0	PCP_CMPLTDATABASE	1 when client has completed latest database. 0 if latest database not completed or still being captured.
1	PCP_STATUSDATABASE	Database status. 1= changed, 0= unchanged. If it is 1 server will make it 0 after responding once with status 1.
2	PCP_OK_TO_POLL	During field-testing set it to 1 to allow poll. During home testing with FieldServer it should be 0 to allow poll.
3	PCP_OK_TO_DUMP	Set to 1 to print database info to file "pcp_db.txt".
4	PCP_STAT_NODE_CREATED_BACNET_IP	Number of BacNet_IP Nodes created in auto Server mode.
5	PCP_STAT_NODE_CREATED_BACNET_ETH	Number of BacNet_ETH Nodes created in auto Server mode.
6	PCP_STAT_RESPONSE_INHIBITED	To stop Server response, set this field to 1. To remove this restriction set this back to 0.
7	PCP_STAT_RESEND_OF_POLL	Number of times client re-polls the node .
8		Obj Id of 1st bacnet object created is stored here
9		Obj Id of 2nd bacnet object created is stored here
10		Obj Id of 3rd bacnet object created is stored here
11		Obj Id of 4th bacnet object created is stored here
12		Obj Id of 5th bacnet object created is stored here
13		Obj type of 1st bacnet object created is stored here
14		Obj type of 2nd bacnet object created is stored here
15		Obj type of 3rd bacnet object created is stored here
16		Obj type of 4th bacnet object created is stored here

Stat Number	Stats	Description
17		Obj type of 5th bacnet object created is stored here
18		When set to 1 the server side will respond reporting all properties as read only
19		Driver sets this to 1 to report that a BACnet MSTP node has been created.
20		Set to 1 to tell driver to print message each time it stores data.
21		Driver sets this to 1 to report that a Metasys node has been created.
22		Count of the number of server objects made whose MAP DESCRIPTOR data type = BI
23		Count of the number of server objects made whose MAP DESCRIPTOR data type = BO
24		Count of the number of server objects made whose MD data type = BV
25		Count of the number of server objects made whose MD data type = AI
26		Count of the number of server objects made whose MD data type = AO
27		Count of the number of server objects made whose MD data type = AV

Appendix C. Reference

Appendix C.1. Command Summary

Additional information can be found in the documented named “FST Protocol Spec - PCP Advanced Driver Spec (FS-8700-109)”

Command	Action	Notes
0x01	Poll Object Class Count	
0x02	Poll Object Class Configuration	
0x03	Poll Object Class Property	
0x04	Poll Object Instance Count	
0x05	Poll Object Instance Info	
0x06	Configuration Valid / Changed	
0x11	Read Property Values	There are limits to this command. Read the protocol specification document.
0x12	Read All Property Values	There are limits to this command. Read the protocol specification document.
0x21	Write Property Value	