Remote Control - MacNet - LabView

Description

The Maccor tester software, MacTest32.exe, can be controlled by other software through its UDP/IP and TCP/IP interfaces. The software to control the Maccor tester can either be running on the same computer by using the IP loop-back address (127.0.0.1) or from another networked computer using the IP address of the tester computer.

The protocol is very open and new features can easily be added without any impact on already present features.

This document describes the interface, and it is up to the user to write software to communicate with the tester computer.

Network interface TCP/IP UDP/IP

TPC/IP and UDP/IP

Both TCP/IP and UDP/IP protocols are offered.

UDP is connection-less and messages sent to the port specified in the System.ini file in

[Misc_Options]
MacNet_UDP_Port=57550

(57550 being the default value for UDP)

will be processed and the reply sent to the IP address they came from.

TCP connections are accepted at the port specified in the System.ini file in

[Misc_Options]
MacNet_TCP_Port=57560

(57560 being the default value for TCP)

If the port numbers are changed, please note that the tester program must be restarted.

Besides the underlying TCP or UDP protocols, the message structures are identical as described in the following topics.

NOTE! Firewalls

If MacNet is used from another computer, the firewall on the tester PC will typically block for incoming network messages including the ones used in the MacNet. So to use TCP/IP the MacNet_TCP_Port number must be opened for incoming messages. For UDP/IP, the MacNet_UDP_Port number and MacNet_UDP_Port+1 must be open on both the tester and remote computer.

In addition the network Ping, aka ICMPv4 is often blocked by the Windows firewall by default and it will have to be enabled on the tester PC's:

- Search for Windows Firewall, and click to open it.
- Click Advanced Settings on the left.
- From the left pane of the resulting window, click Inbound Rules.
- In the right pane, find the rules titled File and Printer Sharing (Echo Request - ICMPv4-In).
- Right-click each rule and choose Enable Rule.

Implementation

A demo application including source code in Delphi is supplied with the Maccor software and is installed in the C:\Maccor\MacNet\Demos folder. It will demonstrate most of the messages. Besides investigating the source code it is also suggested to just run the demo application and investigating the communication with a network analyzer or package sniffer like WireShark.

- **Maccor can only offer support on MacNet communication issues if WireShark pcapng log files of the network messages in question are provided.**

JSON protocol

JSON

In addition to the native UDP and TCP implementation, a JSON version is also offered.

JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute –value pairs. It is the most common data format used for asynchronous browser/server communication, largely replacing XML. For more information about the JSON protocol, please visit these sites:

- http://www.json.org/
- https://tools.ietf.org/html/rfc7159
- https://en.wikipedia.org/wiki/JSON
- http://www.jsonrpc.org/specification

The actual implementation of the JSON messages is described under the individual MacNet functions. In general the values in the JSON implementation are the same as native implementation, except for error messages, booleans, strings in general and dates that are as is or ISO 8601.

JSON requires a TCP connections it is accepted at the port specified in the System.ini file in

[Misc_Options]
MacNet_JSON_Port=57570

(57570 being the default value for JSON/TCP/IP)

If the JSON message itself has no errors and can be processed by the Maccor tester program, it will return the same data as the corresponding regular MacNet message. If problems parsing the JSON message are encountered, it will be reported according to JSON-RPC 2.0 with one of these Error messages:

- Parse error
- Method MacNet, jsonrpc 2.0 or id not found
- Invalid params
- "FClass" key does not exist or value syntax error
- "FNum" key does not exist or value syntax error
- Invalid FClass
- Invalid FNum
- MacNet error
- Missing object
- Illegal value

NOTE!

JSON is derived from Java and it is case sensitive like Java and C.

MacNetLib and LabView

MacNetLib32.dll and MacNetLib64.dll – LabView

In addition to using network protocols directly, a thin encapsulation of the network interface is provided in the form of a DLL, MacNetLib32.dll and MacNetLib64.dll (for 32-bit and 64-bit versions of LabView). The main aim of this library is LabView and some values like time stamps are converted from the Unix standard to LabView standard.

Otherwise it is just handling the network (TCP/IP) initialization and passing the data. Although this is an ordinary DLL, it is aimed and LabView and all other applications should use the network interface directly.

The MacNetLib32.dll and MacNetLib64.dll are installed in the C:\Maccor folder and can be called from here if LabView or the LabView application are running on the tester PC. If LabView or the LabView application are running on another PC, the relevant version of the dll must be copied to this computer.

Demos

The c:\Maccor\MacNet\Demos\LabView\ contains a number of demo VI's demonstrating the use of most of the functions. The c:\Maccor\MacNet\Demos\LabView\DemoLibrary\MacNetLib.lvlib is a library with all the individual call, and should be used to copy and paste from.

Implementation

The best starting point is to carefully use and understand the demos and then either develop your own VI's from them or copy and paste individual dll function implementation from the MacNetLib.lvlib library. The details of the individual functions will be explained in the following sections. Please note that the demos point to the 64-bit version of the dll. If you are using the 32-bit version of LabView, each "Library name or path" has to be changed to: "C:\Maccor\MacNetLib32.dll.

A few things to be aware of:

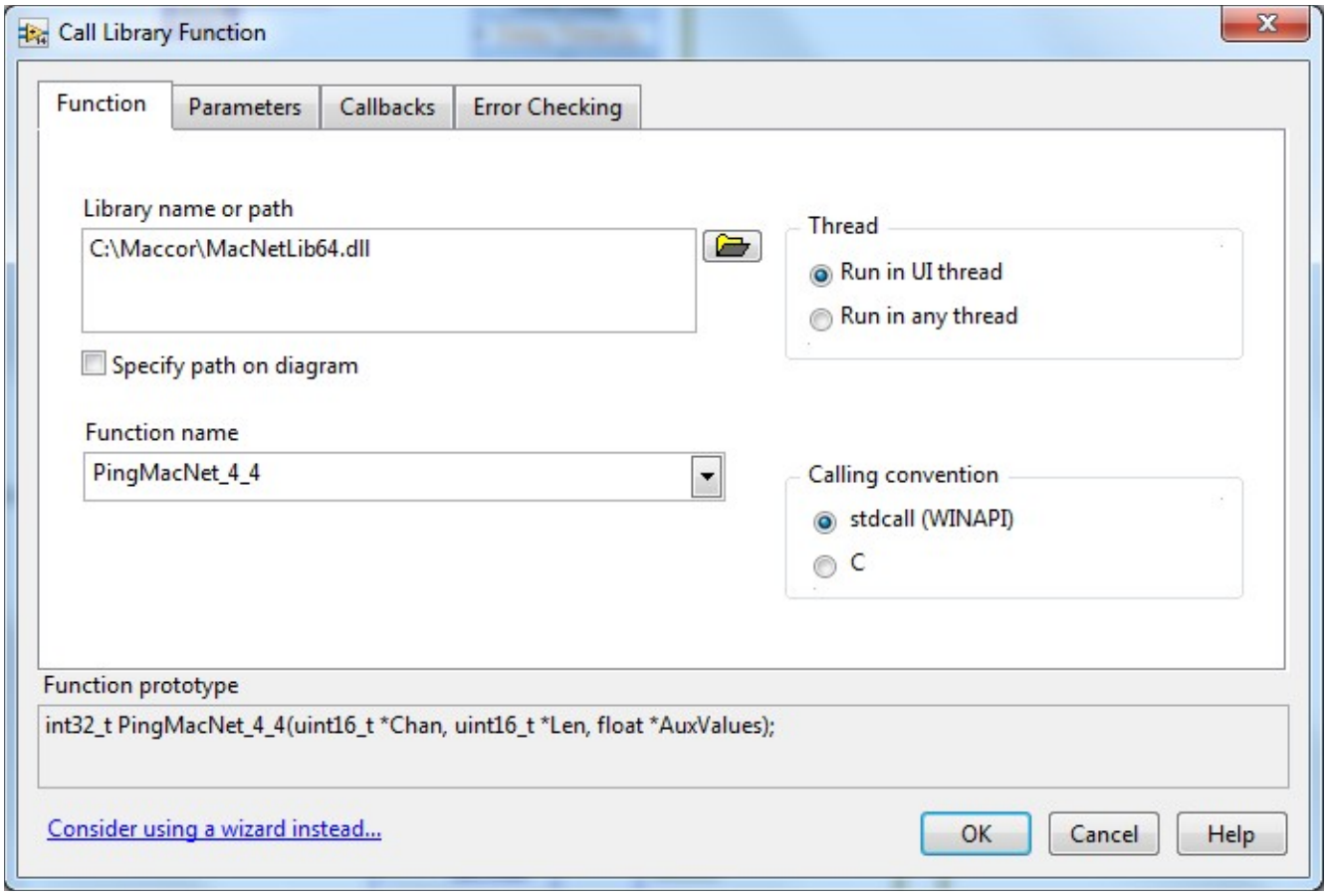
The network interface must be started by calling:

int32_t OpenMacNet(char IPaddress[], uint32_t PortNum);

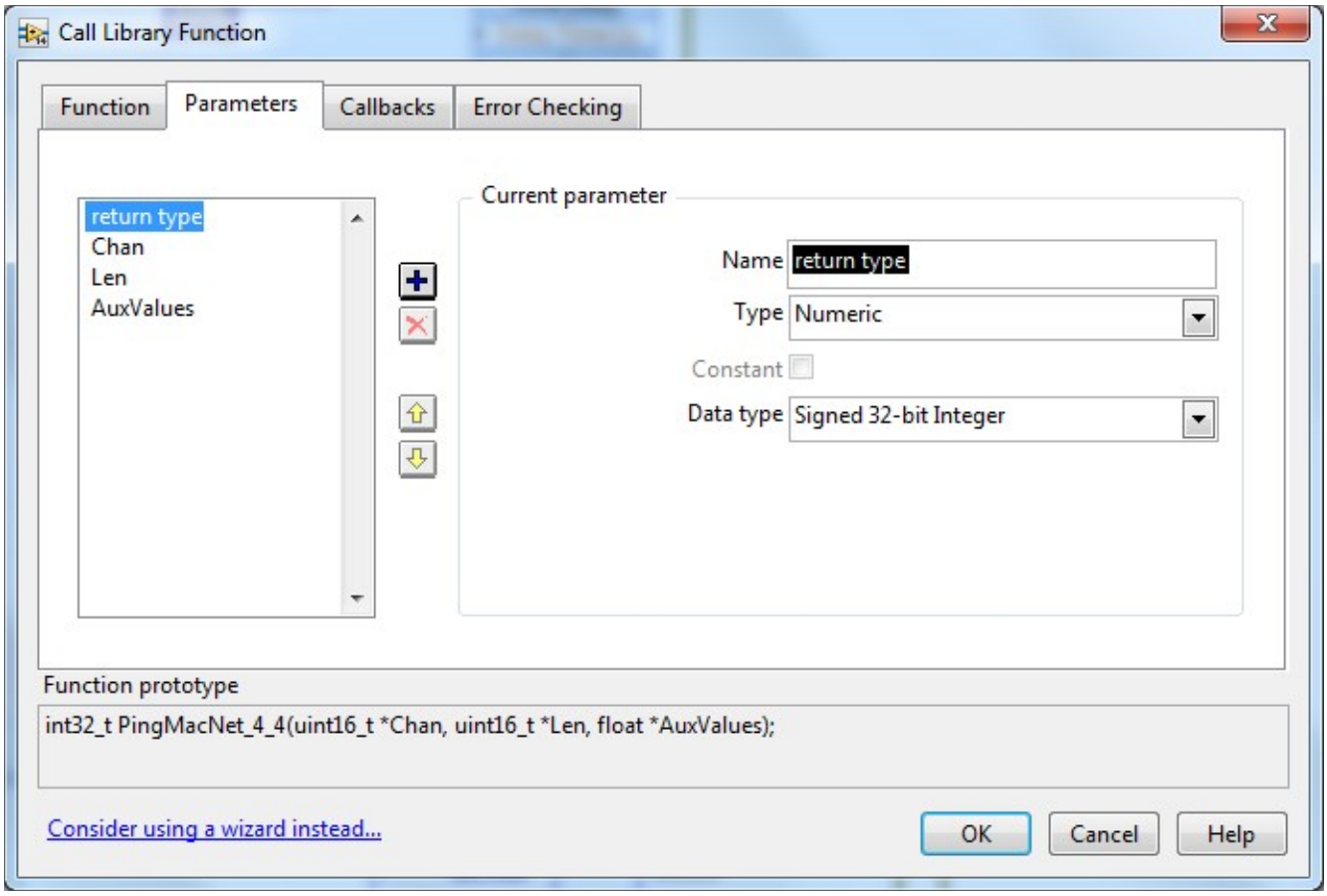
before any other functions and close by:

int32_t CloseMacNet(void);

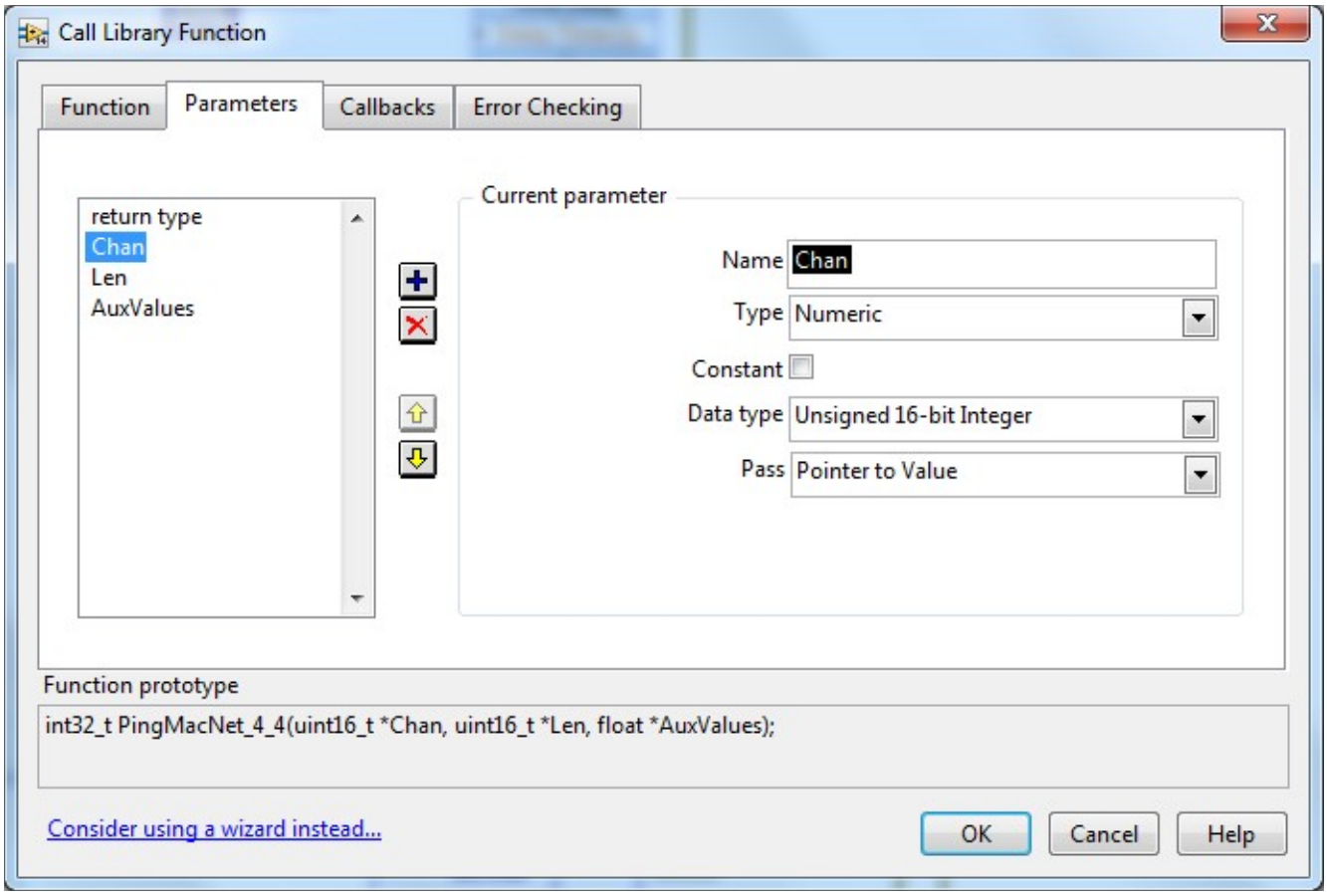
before the program is terminated.



All calls to the MacNetLib must be "Run in the UI thread" and use the "stdcall (WINAPI)" calling convention. If you are using the 32-bit version of LabView, each "Library name or path" has to be changed to: "C:\Maccor\MacNetLib32.dll



There is always returned a 32-bit signed integer. If the operation is successful, the value is 0. Otherwise it is an indication of the type of error.



All values are passed as pointers so they can receive data.

Function prototypes

```
typedef signed char     int8_t;
typedef unsigned char   uint8_t;
typedef short           int16_t;
typedef unsigned short  uint16_t;
typedef int             int32_t;
typedef unsigned int    uint32_t;
typedef unsigned char        *CStr;
```

The return values mean:

- 0: OK
- 1: Socket not open
- 2: Sending message to tester failed
- 3: No reply from tester
- 4: Invalid reply from tester

```
int32_t OpenMacNet(char IPaddress[], uint32_t PortNum);
int32_t CloseMacNet(void);

int32_t PingMacNet_0_0(void);
```

```
int32_t PingMacNet_1_1(uint8_t *ExeMajor, uint8_t *ExeMinor, uint16_t *ExeBuild,
       uint8_t *DllMajor, uint8_t *DllMinor, uint16_t *DllBuild, double *ExeDT,
double *DllDT);

int32_t PingMacNet_1_2(CStr SystemID, uint8_t *SystemType, uint16_t
*ControllerBoards,
       uint16_t *TestChannels, uint16_t *AuxBoards, uint16_t *AuxChannels,
       uint16_t *SMB1Boards, uint16_t *SMB3Boards);


int32_t PingMacNet_3_1(uint16_t *Chan, float TickSpeed, float *Vmax, float *Vmin,
       float *VSafeMax, float *VSafeMin, float *IR4Chg, float *IR3Chg, float *IR2Chg,
       float *IR1Chg, float *IR4Dis, float *IR3Dis, float *IR2Dis, float *IR1Dis,
       float *ISafeChg, float *ISafeDis, float *PBatSafeChg, float *PBatSafeDis,
       float *PChanSafeChg, float *PChanSafeDis, float *PSVChg, float *PSVDis);

int32_t PingMacNet_3_2(uint16_t *Chan, double *VChg, double *VDis, double *IChgR4,
       double *IDisR4, double *IChgR3, double *IDisR3, double *IChgR2,
       double *IDisR2, double *IChgR1, double *IDisR1);


int32_t PingMacNet_4_1(uint16_t *Chan, uint16_t *Len, uint8_t *RF1, uint8_t *RF2,
uint16_t *Stat);

int32_t PingMacNet_4_2(uint16_t *Chan, uint16_t *Len, float *Volts);

int32_t PingMacNet_4_3(uint16_t *Chan, uint16_t *Len, float *Currents);

int32_t PingMacNet_4_4(uint16_t *Chan, uint16_t *Len, float *AuxValues);

int32_t PingMacNet_4_5(uint16_t *Chan, uint16_t *Len, CStr AuxUnits);

int32_t PingMacNet_4_6(uint16_t *Chan, CStr TestName, CStr Comment, CStr ProcName,
CStr ProcDesc);

int32_t PingMacNet_4_7(uint16_t *Chan, uint8_t *RF1, uint8_t *RF2, uint16_t *Stat,
uint32_t *LastRec,
       uint32_t *Cycle, uint16_t *Step, float *TestTime, float *StepTime, float
*Capacity,
       float *Energy, float *Current, float *Voltage, double *TesterTime);

int32_t PingMacNet_4_8(uint16_t *Chan, uint32_t *GlobFlags, float *VARs);


int32_t PingMacNet_5_1(uint32_t *GlobFlags);


int32_t PingMacNet_6_1(uint16_t *Chan);

int32_t PingMacNet_6_2(uint8_t *StartDataType, uint8_t *StartDataVersion, CStr
TestName,
       CStr ProcName, CStr Comment, float *Crate, uint8_t *ChamberNum);

int32_t PingMacNet_6_3(uint16_t *Chan);

int32_t PingMacNet_6_4(uint16_t *Chan);

int32_t PingMacNet_6_5(uint16_t *Chan);
```

```
int32_t PingMacNet_6_6(uint16_t *Chan);

int32_t PingMacNet_6_7(uint16_t *Chan, CStr TestName, float *Current, float *Voltage,
       float *Power, float *Resistance, uint8_t *CurrentRange, uint8_t *ChMode,
       float *DataTime, float *DataV, float *DataI);

int32_t PingMacNet_6_8(uint16_t *Chan, float *Current, float *Voltage, float *Power,
       float *Resistance, uint8_t *CurrentRange, uint8_t *ChMode);

int32_t PingMacNet_6_9(uint16_t *Chan, uint8_t *VarNum, float *VARiable);


int32_t PingMacNet_7_1(uint16_t *Chan, uint8_t *SBbrd, uint8_t *SBPos, uint8_t
*SBType,
       uint8_t *ServerVersionMajor, uint8_t *ServerVersionMinor,
       uint8_t *ClientVersionMajor, uint8_t *ClientVersionMinor,
       uint8_t *DLLVersionMajor, uint8_t *DLLVersionMinor,
       uint16_t *DLLCompileNum, CStr DLLversionString, CStr DLLName,
       uint32_t *SecsLastRead, uint32_t *SecsLastHeader, uint8_t *Status, uint16_t
*BusVolt, int16_t *Temperature);

int32_t PingMacNet_7_4(uint16_t *Chan, uint8_t *RegAddr, uint16_t *Data);
```

MacNet Functions

The functions in the MacNet interface are grouped in classes and each function in a function class has a number, and all messages are identified by these numbers. In addition these numbers are followed by a Chan - channel number, a Len - length and the data. This structure is the same for messages sent to the Maccor program and well as the replies. Function Class and Function No will always be the same in the reply.

Function Class	Word	General message type
Function No	Word	Type with the general message type
Chan	Word	Test channel. 0-based.
Len	Word	Tx: Number of Test channels Rx: Number of bytes of data
Data	“Len number” of bytes	

NOTE!
Chan is 0 (zero) based, so to e.g. address test channel number 4, Chan would be 3.

Data types:

All types are Little-Endian, also known as least-significant byte first and Intel convention.

Byte	8 bits
Char	A byte interpreted as an ASCII character
Word	2 bytes unsigned
UInt16	2 bytes unsigned
Int16	2 bytes signed
Dword	4 bytes unsigned
Int32	4 bytes signed
UInt64	8 bytes unsigned
Single	Single precision IEEE 754 floating point. 4 bytes
Double	Double precision IEEE 754 floating point. 8 bytes

The interpretation of Data is explained in the following sections

Function classes and number overview:

For a Function Class of 0, the received message will be replied as is, i.e. an echo function. This can be convenient for initial debugging.

1. System info
    1. Version info
    2. General system information
2. Controller board info
3. Channel info
    1. Channel specification
    2. Calibration dates
4. Channel status
    1. Channel status of multiple channels
    2. Voltage readings of multiple channels
    3. Current readings of multiple channels
    4. Auxiliary input readings
    5. Auxiliary input engineering units
    6. File name, test procedure name and comments
    7. All status and readings of one channel
    8. Global flags and VARs
5. System commands
6. Channel commands
    1. Select/unselect
    2. Start test on selected channels
    3. Suspend
    4. Resume
    5. Reset
    6. Archive
    7. Start direct mode
    8. Set direct mode output
7. Smart Battery
    1. Version and status info
    2. Suspend smart battery polling
    3. Resume smart battery polling
    4. Get data from a register
    5. Get the scan list
    6. Get raw data
    7. Get the header scan list
    8. Get raw header data
    9. Write multiple bytes
    10. Write multiple bytes
    11. Request read multiple bytes
    12. Return the bytes read
    13. .
    14. .
    15. .
    16. .
    17. .
    18. .
    19. .
    20. Read one entry of scan list
    21. Write one entry of scan list
    22. Clear the scan list
    23. Save the updated scan list

Function Class 0: Echo

For a Function Class of 0, the received message will be replied as is, i.e. an echo function. This can be convenient for initial debugging.

JSON

Transmit message:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 0,
   "FNum": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 0,
   "FNum": 0
 },
 "id": 1987
}
```

Function Class 1: System info

MacNet functions to obtain system info

(1, 1) Version info

The version of the tester software and MacNet interface will be returned.

Transmit message:

Function Class	WORD	1
Function No	WORD	1
Chan	WORD	N/A
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	1
Function No	WORD	1
Chan	WORD	
Len	WORD	
APIVersion	Word	Version of the MacNet interface
MacTest32EXEversion	1 word and 2 bytes	Build number, minor and major version
MacTest32DLLVersion	1 word and 2 bytes	Build number, minor and major version
MacTest32ExeDT	UInt64/Double	Build TimeStamp of MacTest32.exe
MacTest32DLLDT	UInt64/Double	Build TimeStamp of MacTest32.DLL

MacNetLib function prototype:

```
int32_t PingMacNet_1_1(uint8_t *ExeMajor, uint8_t *ExeMinor, uint16_t *ExeBuild,
       uint8_t *DllMajor, uint8_t *DllMinor, uint16_t *DllBuild, double *ExeDT,
double *DllDT);
```

For MacNet directly the TimeStamp is the Unix time format (milliseconds since start of January 1, 1970). For MacNetLib (LabView) the TimeStamp format is the LabView format (amount of seconds from the epoch time of 12 a.m., January 1, 1904)

JSON

Transmit message:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 1
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":1,
   "FNum":1,
   "APIVersion":1,
   "MacTest32EXEversionMajor":3,
   "MacTest32EXEversionMinor":2,
   "MacTest32EXEversionBuild":18,
   "MacTest32DLLversionMajor":3,
   "MacTest32DLLversionMinor":2,
   "MacTest32DLLversionBuild":18,
   "MacTest32ExeDT":"2016-11-08T15:02:58",
   "MacTest32DLLDT":"2016-11-13T11:29:48"
 }
 ,
 "id":1987
}
```

NOTE: The main components of the Maccor battery tester software are the MacTest32.EXE and MacTest32.DLL. They are intimately related and their version numbers must be identical. If they are not, please immediately contact Maccor Customer Service or the Maccor personal who has supplied the software for an update.

(1, 2) General system information

General system information will be returned:

Transmit message:

Function Class	WORD	1
Function No	WORD	2
Chan	WORD	N/A
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	1
Function No	WORD	2
Chan	WORD	N/A
Len	WORD	
SystemID	50 bytes (ASCII)	Name of the test system
System Type	1 byte	0: Lab system 1: 500 ms mode 2: N/A 3: 50 ms mode 4: First generation 8500 5: Second generation 8500
Number of channel controller boards	Word	
Number of test channels	Word	
Number of AUX boards	Word	
Number of AUX inputs	Word	
Number of SMB1 boards	Word	The number of SMB1 positions is 12 per SMB1 board
Number of SMB3 boards	Word	The number of SMB3 positions is 12 per SMB3 board
Channel Number Offset	DWORD	ChannelNumberOffset from System.ini

MacNetLib function prototype:

```
int32_t PingMacNet_1_2(CStr SystemID, uint8_t *SystemType, uint16_t
*ControllerBoards,
       uint16_t *TestChannels, uint16_t *AuxBoards, uint16_t *AuxChannels,
       uint16_t *SMB1Boards, uint16_t *SMB3Boards);
```

JSON

Transmit message:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 2
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":1,
   "FNum":2,
   "SystemID":"Win10",
   "SystemType":0,
   "ControllerBoards":3,
   "TestChannels":12,
   "AuxBoards":1,
   "AuxChannels":128,
   "SMB1Pos":0,
```

```
  "SMB3Pos":1
 }
 ,
 "id":1987
}
```

(1, 3) Procedure Exists

Check if a given test procedure exists:

Transmit message:

Function Class	WORD	1
Function No	WORD	3
Chan	WORD	N/A
Len	WORD	Length of procedure name
ProcName	25 bytes	Up to 25 ascii characters for the test procedure name. The test procedure must exist in the C:\Maccor\Procedur folder and the ".000" should not be part of the name. The field is fixed in length, so remaining characters should be space characters.

MacNetLib function prototype:

```
int32_t PingMacNet_1_3(CStr ProcName, uint8_t *ProcedurExists);
```

Reply message:

Function Class	WORD	1
Function No	WORD	2
Chan	WORD	N/A
Len	WORD	
ProcedurExists	1 byte	0: Procedure does not exist 1: Procedure exists

JSON

Transmit message:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 3,
   "ProcName": "Procedure Name"
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
```

```
  "FClass":1,
  "FNum":3,
  "ProcedurExists":"TRUE"
 }
 ,
 "id":1987
}
```


(1, 4) Test File Name Exists

Check if a given test file name exists in the Archive folder:

Transmit message:

Function Class	WORD	1
Function No	WORD	4
Chan	WORD	N/A
Len	WORD	Length of TestName
TestName	25 bytes	Up to 25 ascii characters for the data file name. The field is fixed in length, so remaining characters should be space characters.

Reply message:

Function Class	WORD	1
Function No	WORD	2
Chan	WORD	N/A
Len	WORD	
TestNameExists	1 byte	0: Test name does not exist 1: Test name exists

JSON

Transmit message:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 4,
   "TestName": "Test Name"
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":1,
   "FNum":4,
   "TestNameExists":"TRUE"
 }
```

,
"id":1987
}

(1, 5) Get File Listing

Get a list of file names in different folders on the tester.

Transmit message:

Function Class	WORD	1
Function No	WORD	5
Chan	WORD	N/A
Len	WORD	2
File Type	1 byte	1: Active (numerical extension only, i.e. data files) 2: Archive (numerical extension only, i.e. data files) 3: Procedur (*.000 and *.999) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF)
Command	1 byte	0: Build file listing 1: Request next file name

Reply messages:

For command =0:

Function Class	WORD	1
Function No	WORD	5
Chan	WORD	N/A
Len	WORD	3
File type	1 byte	Same as in the send message
Command	1 byte	0
Number of files	UInt16	Number of files

For command =1:

Function Class	WORD	1
Function No	WORD	5
Chan	WORD	N/A
Len	WORD	
File type	1 byte	Same as in the send message
Command	1 byte	1
Number of files	UInt16	Number of files
Index of current file	UInt16	
File Date	Int64/Double	TimeStamp
File Size	Int64	Size of the file
File name length	UInt16	Length of the file name including extension
File name	"File name length" bytes	ASCII

For MacNet directly the TimeStamp is the Unix time format (milliseconds since start of January 1, 1970). For MacNetLib (LabView) the TimeStamp format is the LabView format (amount of seconds from the epoch time of 12 a.m., January 1, 1904)

JSON

Transmit message:


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 5,
   "FileType": 2,
   "Command": 0
 },
 "id": 1987
}
```


and


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 1,
   "FNum": 5,
   "FileType": 2,
   "Command": 1
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":1,
   "FNum":5,
   "FileType":2,
   "Command":0,
   "NumberOfFiles":282
 }
 ,
 "id":1987
}
```


and


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":1,
   "FNum":5,
   "FileType":2,
   "Command":1,
   "NumberOfFiles":282,
   "Index":9,
   "FileDate":"2016-08-16T11:43:03",
```

```
  "Size":17157,
  "FileName":"4259-85-2853_001.005"
 }
 ,
 "id":1987
}
```


(1, 6) Send file to tester

The send file to tester is inspired by the TFTP protocol but not completely compatible. The transmission is initiated by sending an OpCode 2 message to the tester program holding the type of file and the name of it. If the file can be created, the tester program will acknowledge this with an OpCode of 4 including a block number of 0. If the file cannot be created, an error code with an OpCode of 5 will be sent. Successive and numbered 500 byte block of data will then have to be sent with an OpCode of 3 and they are acknowledged with OpCode 4 and the same block number. Transmission completes by sending a data block smaller than 500.

General transmit message:

Function Class	WORD	1
Function No	WORD	6
Chan	WORD	N/A
Len	WORD	2
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF)
Command/OpCode	1 byte	2: Write request (WRQ) 3: Data (DATA) 4: Acknowledgment (ACK) 5: Error (ERROR)

OpCode 2: Write request (WRQ) transmit message:

Function Class	WORD	1
Function No	WORD	6
Chan	WORD	N/A
Len	WORD	4 + File Name Length
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF)
Command/OpCode	1 byte	2: Write request (WRQ)
File name length	UInt16	Length of the file name including extension
File name	"File name length" bytes	ASCII

OpCode 4: Acknowledgment (ACK) message:

Function Class	WORD	1
Function No	WORD	6
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab)

		7: Waveforms (*.MWF)
Command/OpCode	1 byte	4: Acknowledgment (ACK)
BlockNo	WORD	Block received

OpCode 3: Data (DATA) message:

Function Class	WORD	1
Function No	WORD	6
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF)
Command/OpCode	1 byte	3: Data (DATA)
BlockNo	WORD	Block no
Data	500 bytes	500 bytes of data from the file.

OpCode 5: Error (ERROR) message:

Function Class	WORD	1
Function No	WORD	6
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF)
Command/OpCode	1 byte	5: Error (ERROR)
ErrorCode	WORD	Block no
Error message	Null terminated ASCII string	

JSON

JSON does not support binary data so this function is not implemented in JSON.

(1, 7) Get file from tester

The get file from tester is inspired by the TFTP protocol but not completely compatible. The transmission is initiated by sending an OpCode 1 message to the tester program holding the type of file and the name of it. If the file exists, the tester program will reply with the first data block 1 with an OpCode of 3. If the file does not exist, an error code with an OpCode of 5 will be sent. Acknowledging the data wit message OpCode 4 and the same block number will cause the next data blck to be sent. Transmission completes when a data block smaller than 500.

Please note that although data files can be transferred from the tester by this method, it is slow. Data files should be handled by the MIMS.

General message:

Function Class	WORD	1
Function No	WORD	7
Chan	WORD	N/A
Len	WORD	2

File type	1 byte	1: Active (numerical extension only, i.e. data files) 2: Archive (numerical extension only, i.e. data files) 3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF) 8: SysEvent.log
Command/OpCode	1 byte	1: Read request (RRQ) 3: Data (DATA) 4: Acknowledgment (ACK) 5: Error (ERROR)

OpCode 1: Read request (WRQ) transmit message:

Function Class	WORD	1
Function No	WORD	7
Chan	WORD	N/A
Len	WORD	4 + File Name Length
File type	1 byte	1: Active (numerical extension only, i.e. data files) 2: Archive (numerical extension only, i.e. data files) 3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF) 8: SysEvent.log
Command/OpCode	1 byte	1: Read request (RRQ)
File name length	UInt16	Length of the file name including extension
File name	"File name length" bytes	ASCII

OpCode 4: Acknowledgment (ACK) message:

Function Class	WORD	1
Function No	WORD	7
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	1: Active (numerical extension only, i.e. data files) 2: Archive (numerical extension only, i.e. data files) 3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF) 8: SysEvent.log
Command/OpCode	1 byte	4: Acknowledgment (ACK)
BlockNo	WORD	Block received

OpCode 3: Data (DATA) message:

Function Class	WORD	1
Function No	WORD	7
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	1: Active (numerical extension only, i.e. data files) 2: Archive (numerical extension only, i.e. data files) 3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC)

		6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF) 8: SysEvent.log
Command/OpCode	1 byte	3: Data (DATA)
BlockNo	WORD	Block no
Data	500 bytes	500 bytes of data from the file.

OpCode 5: Error (ERROR) message:

Function Class	WORD	1
Function No	WORD	7
Chan	WORD	N/A
Len	WORD	4 File Name Length
File type	1 byte	3: Procedur (*.000) 4: MacUserFunctions (*.DLL) 5: CAN (*.CAN and *.DBC) 6: FRA (*.FRA and *.modulab) 7: Waveforms (*.MWF) 8: SysEvent.log
Command/OpCode	1 byte	5: Error (ERROR)
ErrorCode	WORD	Block no
Error message	Null terminated ASCII string	

JSON

JSON does not support binary data so this function is not implemented in JSON.

Function Class 2: Controller board info

MacNet functions to obtain controller board info

Function Class 3: Channel info

MacNet functions to obtain channel info

(3, 1) Channel specification

The specification of the channel given in Chan will be returned. Up to 128 channels can be requested at the time. For LabView and JSON only the first channel is returned:

Transmit message:

Function Class	WORD	3
Function No	WORD	1
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Data	N/A	

Reply message:

Function Class	WORD	3
Function No	WORD	1
Chan	WORD	
Len	WORD	
TickSpeed	Single	Tick speed of the controller board in ms
Vmax	Single	Max channel voltage
Vmin	Single	Min channel voltage
VSafeMax	Single	Max safety voltage
VSafeMin	Single	Min safety voltage
IR4Chg	Single	Max charge current in range 4

IR3Chg	Single	Max charge current in range 3
IR2Chg	Single	Max charge current in range 2
IR1Chg	Single	Max charge current in range 1
IR4Dis	Single	Max discharge current in range 4
IR3Dis	Single	Max discharge current in range 3
IR2Dis	Single	Max discharge current in range 2
IR1Dis	Single	Max discharge current in range 1
ISafeChg	Single	Max safety current in charge
ISafeDis	Single	Max safety current in discharge
PBatSafeChg	Single	Max battery power in charge
PBatSafeDis	Single	Max battery power in discharge
PChanSafeChg	Single	Max channel power in charge
PChanSafeDis	Single	Max channel power in discharge
PSVChg	Single	Charge power supply voltage
PSVDis	Single	Discharge power supply voltage
		....Data for more channels

MacNetLib function prototype:

```
int32_t PingMacNet_3_1(uint16_t *Chan, float TickSpeed, float *Vmax, float *Vmin,
       float *VSafeMax, float *VSafeMin, float *IR4Chg, float *IR3Chg, float *IR2Chg,
       float *IR1Chg, float *IR4Dis, float *IR3Dis, float *IR2Dis, float *IR1Dis,
       float *ISafeChg, float *ISafeDis, float *PBatSafeChg, float *PBatSafeDis,
       float *PChanSafeChg, float *PChanSafeDis, float *PSVChg, float *PSVDis);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 3,
   "FNum": 1,
   "Chan": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":3,
   "FNum":1,
   "Chan":0,
   "TickSpeed":0.00499999988824129,
   "Vmax":10,
   "Vmin":0,
   "VSafeMax":10,
   "VSafeMin":0,
   "IR4Chg":5,
   "IR3Chg":0.150000005960464,
   "IR2Chg":0.00499999988824129,
   "IR1Chg":0.000150000007124618,
```

```
  "IR4Dis":5,
  "IR3Dis":0.150000005960464,
  "IR2Dis":0.00499999988824129,
  "IR1Dis":0.000150000007124618,
  "ISafeChg":5,
  "ISafeDis":5,
  "PBatSafeChg":50,
  "PBatSafeDis":50,
  "PChanSafeChg":0,
  "PChanSafeDis":0,
  "PSVChg":0,
  "PSVDis":0
 }
 ,
 "id":1987
}
```


(3, 2) Calibration dates

The test channel calibration verification dates of the channel given in Chan will be returned.. This is the information stored in the

C:\Maccor\System\"Name of the system"\Voltage Verify Log.txt
and
C:\Maccor\System\"Name of the system"\Current Verify Log.txt

Transmit message:

Function Class	WORD	3
Function No	WORD	2
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	3
Function No	WORD	2
Chan	WORD	
Len	WORD	
Voltage Charge	UInt64/Double	TimeStamp
Voltage Discharge	UInt64/Double	TimeStamp
Current Charge Range 4	UInt64/Double	TimeStamp
Current Discharge Range 4	UInt64/Double	TimeStamp
Current Charge Range 3	UInt64/Double	TimeStamp
Current Discharge Range 3	UInt64/Double	TimeStamp
Current Charge Range 2	UInt64/Double	TimeStamp
Current Discharge Range 2	UInt64/Double	TimeStamp
Current Charge	UInt64/Double	TimeStamp

Range 1		
Current Discharge Range 1	UInt64/Double	TimeStamp

MacNetLib function prototype:

```
int32_t PingMacNet_3_2(uint16_t *Chan, double *VChg, double *VDis, double *IChgR4,
       double *IDisR4, double *IChgR3, double *IDisR3, double *IChgR2,
       double *IDisR2, double *IChgR1, double *IDisR1);
```

For MacNet directly the TimeStamp is the Unix time format (milliseconds since start of January 1, 1970). For MacNetLib (LabView) the TimeStamp format is the LabView format (amount of seconds from the epoch time of 12 a.m., January 1, 1904)

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 3,
   "FNum": 2,
   "Chan": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":3,
   "FNum":2,
   "Chan":0,
   "VChg":"1970-01-01T00:00:00",
   "VDis":"1970-01-01T00:00:00",
   "IChgR4":"1970-01-01T00:00:00",
   "IDisR4":"1970-01-01T00:00:00",
   "IChgR3":"1970-01-01T00:00:00",
   "IDisR3":"1970-01-01T00:00:00",
   "IChgR2":"1970-01-01T00:00:00",
   "IDisR2":"1970-01-01T00:00:00",
   "IChgR1":"1970-01-01T00:00:00",
   "IDisR1":"1970-01-01T00:00:00"
 }
 ,
 "id":1987
}
```

Function Class 4: Channel status

MacNet functions to read channel status

(4, 1) Channel status of multiple channels

The status of “Len” channels starting from “Chan” will be returned. Four bytes per requested channel. Up to 128 channels can be requested in one message.

Transmit message:

Function Class	WORD	4
Function No	WORD	1
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	1
Chan	WORD	
Len	WORD	
RF1	Byte	Codes coming directly from the controller board 0: Available 1: Charge 2: Dischrge 3: AdvCycle 4: Rest 5: Pause 7: End 8: Ext Chg 9: Ext Dis 19: Pls Chg 20: Pls Dis 21: I/O Out 22: EnvChmbr 23: Scan 26: SubRout and FRA 29: Problem 30: Suspended 31: Complete
RF2	Byte	Codes coming directly from the controller board 0: Start of Step 1: Step Time 4: Current increment 5: Voltage increment 8: Amp Hour 9: Watt Hour 17: Pause Step 18: Auxiliary Input 33: Pulse low 34: Pulse high 37: Digital Input 128: None 129: Step time 130: Test time 131: Cycle Number 132: Current

		133: Voltage 134: Power 135: Resistance 136: Amp Hour 137: Watt Hour 138: Half Cycle Ahr 139: Half Cycle Whr 140: Previous Ahr 141: Previous Whr 142: First Half Cycle Ahr 143: First Half Cycle Whr 144: Last Half Cycle Ahr 145: Last Half Cycle Whr 146: External 147: Loop Count 150: Voltage/Hour 151: Ahr Previous 152: Half Cycle AHr 153: Half Cycle Time 154: Delta I 155: Delta V 157: Aux Volt 158: Thermcpl 159: Thermstr 160: Pressure or Procedure complete when Advance Started 161: Max-Deviation 162: Mean-Deviation 163: SMBus alarm (SMB smart battery) 164: Function 165: Digital Input 189: Safety Capacity 190: Safety Voltage <pk 191: Safety overcharge 192: S (operator suspended or Pause step) 193: O (normal end) 194: P (timeout fault) 195: P (timeout fault) 196: P (overcharge safety) 197: P (voltage safety <pk) 198: P (slave error) 199: Operator forced step 252: P (Slave Current control error) 253: P (V safety absolute) 254: P (lost current control) 255: P (Buffer full)
Stat	Word	The state of the channel in the software 0: Available 1: Selected 2: Active 3: Suspended 4: Completed 5: Problem 6: Not/Avl 7: Reset 8: Start 9: PWR Fail 10: No Cntllr 11: ShutDown

		12: Starting 13: Blocked 14: Waiting 15: FRA 16: Rem Maint 17: Not USED 18: N/A

MacNetLib function prototype:

```
int32_t PingMacNet_4_1(uint16_t *Chan, uint16_t *Len, uint8_t *RF1, uint8_t *RF2,
uint16_t *Stat);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 1,
   "Chan": 4,
   "Len": 2
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":1,
   "Chan":4,
   "Len":2,
   "Status":
   [
     {
       "RF1":0,
       "RF2":0,
       "Stat":0
     }
     ,
     {
       "RF1":0,
       "RF2":0,
       "Stat":0
     }
   ]
 }
 ,
 "id":1987
}
```

(4, 2) Voltage readings of multiple channels

The the actual voltage of “Len” channels starting from “Chan” will be returned as single precision floating point numbers. Four bytes per channel. Up to 128 channels can be requested at the time.

Transmit message:

Function Class	WORD	4
Function No	WORD	2
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	2
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Voltage(Chan)	Single	
Voltage(Chan+1)	Single	
Voltage(Chan+2)	Single	
Voltage(Chan+..)	Single	

MacNetLib function prototype:

```
int32_t PingMacNet_4_2(uint16_t *Chan, uint16_t *Len, float *Volts);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 2,
   "Chan": 0,
   "Len": 8
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":2,
   "Chan":0,
```

```
  "Len":8,
  "Voltage":
  [
    0.539101243019104,
    0,
    0.000152590218931437,
    0.00640878919512033,
    0.378957808017731,
    -1.81513690948486,
    -1.60807204246521,
    -0.922407865524292
  ]
 }
 ,
 "id":1987
}
```


(4, 3) Current readings of multiple channels

The the actual current of “Len” channels starting from “Chan” will be returned as single precision floating point numbers. Four bytes per channel. Up to 128 channels can be requested at the time.

Transmit message:

Function Class	WORD	4
Function No	WORD	3
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	3
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Current(Chan)	Single	
Current(Chan+1)	Single	
Current(Chan+2)	Single	
Current(Chan+..)	Single	

MacNetLib function prototype:

```
int32_t PingMacNet_4_3(uint16_t *Chan, uint16_t *Len, float *Currents);
```

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
```

```
  "FNum": 3,
  "Chan": 0,
  "Len": 8
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
  "FClass":4,
  "FNum":3,
  "Chan":0,
  "Len":8,
  "Current":
  [
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0
  ]
 }
 ,
 "id":1987
}
```


(4, 4) Auxiliary input readings

The the actual readings of the auxiliary channels assigned “Chan” channel will be returned as single precision floating point numbers. Four bytes per auxiliary channel.

Transmit message:

Function Class	WORD	4
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of Aux positions
AuxValue	Single	
AuxValue	Single	
AuxValue	Single	
AuxValue	Single	

MacNetLib function prototype:

```
int32_t PingMacNet_4_4(uint16_t *Chan, uint16_t *Len, float *AuxValues);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 4,
   "Chan": 1
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":4,
   "Chan":1,
   "Len":8,
   "AuxValues":
   [
     69.4158401489258,
     -0.10656064003706,
     -0.0839281156659126,
     -0.0514887496829033,
     -0.0841225311160088,
     -0.0735516026616096,
     -0.0876320600509644,
     -0.08390723913908
   ]
 }
 ,
 "id":1987
}
```

(4, 5) Auxiliary input engineering units

The the engineering units of the auxiliary channels assigned “Chan” channel will be returned as ASCII characters. Four bytes/characters per auxiliary channel.

Transmit message:

Function Class	WORD	4
Function No	WORD	5
Chan	WORD	Test channel. 0-based.

Len	WORD	
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	5
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of Aux positions
AuxUnit	4 chars	
AuxUnit	4 chars	
AuxUnit	4 chars	
AuxUnit	4 chars	

MacNetLib function prototype:

```
int32_t PingMacNet_4_5(uint16_t *Chan, uint16_t *Len, CStr AuxUnits);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 5,
   "Chan": 1
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":5,
   "Chan":1,
   "Len":8,
   "AuxUnit":
   [
     "C",
     "V",
     "V",
     "V",
     "V",
     "V",
     "V",
     "V"
   ]
 }
 ,
 "id":1987
```

}

(4, 6) File name, test procedure name and comments

The data file name, test comment, procedure name and procedure description of “Chan” channel will be returned:

Transmit message:

Function Class	WORD	4
Function No	WORD	6
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	6
Chan	WORD	
Len	WORD	
TestName	25 bytes	25 ascii characters for the data file name.The field is fixed in length and remaining characters will be space characters.
Comment	80 bytes	80 characters for the test comment. The field is fixed in length and remaining characters will be space characters.
ProcName	25 bytes	25 ASCII characters for the procedure name.The field is fixed in length and remaining characters will be space characters.
ProcDesc	80 bytes	80 characters for the procedure description.The field is fixed in length and remaining characters will be space characters.

MacNetLib function prototype:

```
int32_t PingMacNet_4_6(uint16_t *Chan, CStr TestName, CStr Comment, CStr ProcName,
CStr ProcDesc);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 6,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
```

```
  "FNum":6,
  "Chan":3,
  "TestName":"Arbitrary file name 2016-11-10 10-07-05_126",
  "ProcName":"Send email"
 }
 ,
 "id":1987
}
```


(4, 7) All status and readings of one channel

The status and readings of “Chan” channel will be returned.

Transmit message:

Function Class	WORD	4
Function No	WORD	7
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	7
Chan	WORD	
Len	WORD	
RF1	Byte	See function (4, 1) for details
RF2	Byte	See function (4, 1) for details
Stat	Word	See function (4, 1) for details
Last record num	Dword	
Cycle	Dword	
Step	Word	
Test time	Single	S
Step time	Single	S
Capacity	Single	Ah
Energy	Single	Wh
Current	Single	A
Voltage	Single	V
Tester PC time	UInt64/Double	TimeStamp

MacNetLib function prototype:

```
int32_t PingMacNet_4_7(uint16_t *Chan, uint8_t *RF1, uint8_t *RF2, uint16_t *Stat,
uint32_t *LastRec,
       uint32_t *Cycle, uint16_t *Step, float *TestTime, float *StepTime, float
*Capacity,
       float *Energy, float *Current, float *Voltage, double *TesterTime);
```

For MacNet directly the TimeStamp is the Unix time format (milliseconds since start of January 1, 1970). For MacNetLib (LabView) the TimeStamp format is the LabView format (amount of seconds from the epoch time of 12 a.m., January 1, 1904)

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 7,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":7,
   "Chan":3,
   "RF1":31,
   "RF2":193,
   "Stat":4,
   "LastRecNum":18,
   "Cycle":0,
   "Step":2,
   "TestTime":15,
   "StepTime":10,
   "Capacity":0,
   "Energy":0,
   "Current":0,
   "Voltage":0.0062561989761889,
   "TesterTime":"2016-11-14T09:24:08"
 }
 ,
 "id":1987
}
```

(4, 8) Global flags and VARs

The Global flags and VARs of “Chan” channel will be returned:

Transmit message:

Function Class	WORD	4
Function No	WORD	8
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	4

Function No	WORD	8
Chan	WORD	
Len	WORD	
Global flags	Dword	
VAR1 - VAR15	15 single	

MacNetLib function prototype:

```
int32_t PingMacNet_4_8(uint16_t *Chan, uint32_t *GlobFlags, float *VARs);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 8,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":8,
   "Chan":3,
   "Chan":3,
   "GlobFlags":"0x00000020",
   "VARs":
   [
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
     0
   ]
 }
 ,
 "id":1987
}
```

(4, 9) Test time of multiple channels

The the test time in seconds of “Len” channels starting from “Chan” will be returned as single precision floating point numbers. Four bytes per channel. Up to 128 channels can be requested at the time.

Transmit message:

Function Class	WORD	4
Function No	WORD	9
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	2
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of channels
Test time(Chan)	Single	
Test time(Chan+1)	Single	
Test time(Chan+2)	Single	
Test time(Chan+..)	Single	

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 9,
   "Chan": 0,
 "Len": 8
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":9,
   "Chan":0,
   "Len":8,
   "TestTimes":
```

```
  [
    0,
    0,
    0,
    15,
    0,
    0,
    0,
    0
  ]
 }
 ,
 "id":1987
}
```


(4, 10) Get End status

End status of “Chan” channel will be returned. This is a feature commonly used in highly automated PASS/FAIL tests. Typically two end steps are entered in the test procedure and the end conditions will cause the test to GOTO the first and on PASS and the second on FAIL. More End steps can be used as a more detailed grading.

If the test is not in a complete state EndNum is 0

Transmit message:

Function Class	WORD	4
Function No	WORD	10
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	8
Chan	WORD	
Len	WORD	
NumOfEnds	WORD	Total number of end steps in the test procedure
EndNum	WORD	Index of the actual end step

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 4,
   "FNum": 10,
   "Chan": 3
 },
 "id": 1987
}
```


Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":4,
   "FNum":10,
   "Chan":3,
   "NumOfEnds":2,
   "EndNum":1
 }
 ,
 "id":1987
}
```


(4, 11) File name, test procedure name and comments Extended

The data file name, test comment, procedure name and procedure description of “Chan” channel will be returned:

Transmit message:

Function Class	WORD	4
Function No	WORD	6
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	4
Function No	WORD	6
Chan	WORD	
Len	WORD	
TestNameLen	BYTE	Number of characters in TestName.
TestName	TestNameLen	TestNameLen ASCII characters for the data file name.
CommentLen	BYTE	Number of characters in Comment
Comment	CommentLen	CommentLen ASCII characters for the test comment.
ProcNameLen	BYTE	Number of characters in ProcName.
ProcName	ProcNameLen	ProcNameLen ASCII characters for the procedure name.
ProcDescLen	BYTE	Number of characters in ProcDesc.
ProcDesc	ProcDescLen	ProcDescLen characters for the procedure description.

MacNetLib function prototype. Use (4, 6).

JSON Use (4, 6).

Function Class 5: System commands

(5, 1) Set Global Flags

Set the Global Flags to the dword (32-bit unsiged int) of the first four bytes in Data.

Transmit message:

Function Class	WORD	5
Function No	WORD	1
Chan	WORD	

Len	WORD	4
GlobFlags	DWORD	Global Flags

MacNetLib function prototype:

```
int32_t PingMacNet_5_1(uint32_t *GlobFlags);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 5,
   "FNum": 1,
   "GlobFlags": "0x00000001"
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":5,
   "FNum":1,
   "New GlobFlags":"0x00000001"
 }
 ,
 "id":1987
}
```

Function Class 6: Channel commands

MacNet functions to control individual test channels

(6, 1) Select/unselect

Selects the test channels to be started:

Transmit message:

Function Class	WORD	6
Function No	WORD	1
Chan	WORD	Works in two ways: **1.** If Chan is set to 65535, then each byte in “Data” represent a bitmap of 8 channels. If the bit is 1, the channel is selected (if it is in a available state) and if it is 0 the channel and the channel is selected the channel becomes available. E.g. if the first three bytes are 01, 05, 02, then channel 1, 9, 11, 26 will be selected (if possible) and any other selected channel be dis-selected. **NOTE: Channels numbers are 1 based** **2.** Alternatively, if Chan is smaller than 65535, then the Chan will be selected if it is in an available state. If Chan is already in a selected

		state, then it will become available. Equivalent of clicking on the channel in the tester program.
Len	WORD	Number of channels divided by 8
Data	N/A	

MacNetLib function prototype:

```
int32_t PingMacNet_6_1(uint16_t *Chan);
```

NOTE!
In MacNetLib.dll, only option 2 is supported.

JSON

Transmit message (Example). JSON is implemented for option 2 only:

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 1,
   "Chans": [1, 2, 4, 5]
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":1,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 2) Start test

Starts selected test channels with these start data in the “Data” area:

Start Type 1
Transmit message:

Function Class	WORD	6
Function No	WORD	2
Chan	WORD	N/A
Len	WORD	137
StartDataType	Byte	1
StartDataVersion	Byte	1. Reserved for future use

TestName	25 bytes	Up to 25 ascii characters for the data file name. “Random” will generate a pseudo-random name. The field is fixed in length, so remaining characters should be space characters.
ProcName	25 bytes	Up to 25 ascii characters for the test procedure name. The test procedure must exist in the C:\Maccor\Procedur folder and the “.000” should not be part of the name. The field is fixed in length, so remaining characters should be space characters.
Comment	80 bytes	Up to 80 characters for the test comment. The field is fixed in length, so remaining characters should be space characters.
Crate	Single	C-rate. If C-rate is not used, enter 1
ChamberNum	Byte	Environmental chamber number. If the environmental chamber is not used, enter 0.

MacNetLib function prototype:

```
int32_t PingMacNet_6_2(uint8_t *StartDataType, uint8_t *StartDataVersion, CStr
TestName,
       CStr ProcName, CStr Comment, float *Crate, uint8_t *ChamberNum);
```

Start Type 2
Transmit message:

Function Class	WORD	6
Function No	WORD	2
Chan	WORD	0xFFFF: Start selected channel(s) Legal channel no.: Start this channel immediately.
Len	WORD	186
StartDataType	Byte	2
StartDataVersion	Byte	Version 2 and higher support Regimes
TestName	25 bytes	Up to 25 ascii characters for the data file name. “Random” will generate a pseudo-random name. The field is fixed in length, so remaining characters should be space characters.
ProcName	25 bytes	Up to 25 ascii characters for the test procedure name. The test procedure must exist in the C:\Maccor\Procedur folder and the “.000” should not be part of the name. The field is fixed in length, so remaining characters should be space characters.
Comment	80 bytes	Up to 80 characters for the test comment. The field is fixed in length, so remaining characters should be space characters.
Crate	Single	C-rate. If C-rate is not used, enter 1
ChamberNum	Byte	Environmental chamber number. If the environmental chamber is not used, enter 0.
StartCycle	WORD	Default: 0; See Start Test Setup for further details
TotCycles	WORD	Default: 0; See Start Test Setup for further details
Mass	Single	Default: 1; See Start Test Setup for further details
VGain	Byte	Default: 0; Used to change the gain of the constant voltage feedback loop. Only use it after consulting Customer Service. 0: Gain x1 1: Gain x2 2: Gain x4 3: Gain x8 4: Gain /1 5: Gain /2 6: Gain /4 7: Gain /8
AbsTRepAlign	Byte	Default: 0; 1: Use absolute time report alignment. See Start Test Setup for further details
ParallelR	Single	See Start Test Setup for further details
VDivHiR	Single	See Start Test Setup for further details
VDivLoR	Single	See Start Test Setup for further details

CANpos	Signed 2 byte	Default: -1: Not in use.
CANprof	25 bytes	Up to 25 ascii characters for the CAN profile name.
RegimeName	25 bytes	Up to 25 ascii characters for the Regime name. Only functional if **StartDataVersion**>=2 and **Chan** is a legal channel no.

A number will be returned describing the success or failure of the command:

Result	Word	0: OK 10: Cannot start regimes with more than one channel selected. 12: Advanced start is not compatible with regimes. 14: No test procedure was selected, please select a test procedure 15: Channel not Active. Jump start impossible 16: Channel not Selected. Advanced start impossible 17: Channel not Suspended, Cannot Restart 18: The maximum length of the data file name is 256 characters. 19: Name is not a unique filename. 20: Name is not a unique filename 21: EV Chamber in use 22: Invalid entry. 23: Channel in use 24: No Channels were selected to be started. 25: EV Chamber in use.

It is highly recommended to use the 6,11 query (check test start on channel) first to make sure the procedure can run on the desired channels.

JSON

This is essentially type 2 with the addition that the test name and procedure name can be up to 250 characters long. Also, do not select the channel in advance of starting the test. The JSON Start test command will both select and start the channel. If, however, multiple channels should be started with the same test, do first select the channels, then start the test using 65535, meaning "all selected".

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 2,
   "Chan": 3,
   "TestName": "Random",
   "ProcName": "Procedure Name",
   "Comment": "Test comment",
   "Crate": 1,
   "ChamberNum": 0,
   "StartCycle": 0,
   "TotCycles": 0,
   "Mass": 1,
   "VGain": 0,
   "AbsTRepAlign": 0,
   "ParallelR": 0,
   "VDivHiR": 0,
   "VDivLoR": 0,
   "CANpos": -1,
   "CANprof": "",
   "RegimeName": ""
 },
```

```
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":2,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 3) Suspend

Suspend “Chan” test channel if it is in a state where it can be suspended.

Transmit message:

Function Class	WORD	6
Function No	WORD	3
Chan	WORD	Test channel to be suspended. 0-based.
Len	WORD	0
Data	N/A	

MacNetLib function prototype:

```
int32_t PingMacNet_6_3(uint16_t *Chan);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 3,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
```

```
  "FClass":6,
  "FNum":3,
  "Chan":3,
  "Result":"OK"
 }
 ,
 "id":1987
}
```


(6, 4) Resume

Resume “Chan” test channel if it is in state where it can be resumed.

Transmit message:

Function Class	WORD	6
Function No	WORD	4
Chan	WORD	Test channel to be resumed. 0-based.
Len	WORD	0
Data	N/A	

MacNetLib function prototype:

```
int32_t PingMacNet_6_4(uint16_t *Chan);
```

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
  "FClass": 6,
  "FNum": 4,
  "Chan": 3
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
  "FClass":6,
  "FNum":4,
  "Chan":3,
  "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 5) Reset

Reset “Chan” test channel.

Transmit message:

Function Class	WORD	6
Function No	WORD	5
Chan	WORD	Test channel to be reset. 0-based.
Len	WORD	0
Data	N/A	

MacNetLib function prototype:

```
int32_t PingMacNet_6_5(uint16_t *Chan);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 5,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":5,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 6) Archive

Clear or Archive “Chan” test channel.

Transmit message:

Function Class	WORD	6
Function No	WORD	6
Chan	WORD	Test channel to be archived. 0-based.
Len	WORD	0
Data	N/A	

MacNetLib function prototype:

```
int32_t PingMacNet_6_6(uint16_t *Chan);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 6,
   "Chan": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":6,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 7) Start direct mode

Start direct mode. Initializes direct remote control of the test channel with the following data in the "Data" area. The channel will output this current, voltage and power - mutually limiting as standard – until the values are changed with the "(6, 8) Set direct output" command described below or stopped.

Transmit message:

Function Class	WORD	6
Function No	WORD	7
Chan	WORD	Test channel. 0-based.
Len	WORD	55
TestName	25 bytes	Up to 25 ascii characters for the data file name. "Random" (Case sensitive) will generate a pseudo-random name. The field is fixed in length, so remaining characters should be space characters.
Current	Single	Amps. Mutually limiting. Set outside range to ignore. 0 is within range. Must be active to function in (6, 8) Set direct mode output.
Voltage	Single	Voltage. Mutually limiting. Set outside range to ignore. 0 is within range. Must be active to function in (6, 8) Set direct mode output.
Power	Single	Watts. Mutually limiting. Set outside range to ignore. 0 is within range. Must be active to function in (6, 8) Set direct mode output.
Resistance	Single	

		Ohms. Mutually limiting. Set to 0 to ignore. Must be active to function in (6, 8) Set direct mode output.
Current range	Byte	The desired current range: 1, 2, 3, 4
Charge mode	Byte	67 i.e. 'C' for charge, 68 i.e. 'D' for discharge and 82 i.e. 'R' for rest
Data record time	Single	Time increment between data records in the data file. Enter 0 for no data.
Data record voltage	Single	Voltage increment to generate a data record in the data file. Enter 0 to disable.
Data record current	Single	Current increment to generate a data record in the data file. Enter 0 to disable.

- **Only THREE out of the four (Current, Voltage, Power, Resistance) set points can be used simultaneously.**
- **Set points must be active, i.e. within range, in the (6, 7) Start Direct Mode to be functional in the (6, 8) Set direct mode output.**

A number will be returned describing the success or failure of the start:

Result	Word	0: OK 1: Illegal system type. This feature only works for type 0 2: The channel is not available 3: Failed creating the pseudo test procedure

MacNetLib function prototype:

```
int32_t PingMacNet_6_7(uint16_t *Chan, CStr TestName, float *Current, float *Voltage,
       float *Power, float *Resistance, uint8_t *CurrentRange, uint8_t *ChMode,
       float *DataTime, float *DataV, float *DataI);
```

A watchdog timer can be activated in the Misc_Options section of the System.ini file. Setting the MacNetDirectModeWD to a value higher than 0 will cause a channel under direct mode to suspend if it has not been updated within the specified number of seconds.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 7,
   "Chan": 3,
   "TestName": "Random",
   "Current": 0,
   "Voltage": 20,
   "Power": 50,
   "Resistance": 0,
   "CurrentRange": 4,
   "ChMode": "C",
   "DataTime": 1.0,
   "DataV": 0,
   "DataI": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":7,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 8) Set direct mode output

Set direct output. When a channel has been started in direct mode, the output can be set with this command and these arguments:

Transmit message:

Function Class	WORD	6
Function No	WORD	8
Chan	WORD	Test channel. 0-based.
Len	WORD	18
Current	Single	Amps. Mutually limiting. Set outside range to ignore.
Voltage	Single	Voltage. Mutually limiting. Set outside range to ignore.
Power	Single	Watts. Mutually limiting. Set outside range to ignore.
Resistance	Single	Ohms. Mutually limiting. Set to 0 to ignore.
Current range	Byte	The desired current range: 1, 2, 3, 4
Charge mode	Byte	67 i.e. 'C' for charge, 68 i.e. 'D' for discharge and 82 i.e. 'R' for rest

A number will be returned describing the success or failure of the command:

Result	Word	0: OK 1: Illegal system type. This feature only works for type 0 2: The channel is not active 3: Command sent too fast. There must be at least 100 ms (10 ticks) between commands. 4: Direct mode is not active. 5: Direct mode is not ready yet.

MacNetLib function prototype:

```
int32_t PingMacNet_6_8(uint16_t *Chan, float *Current, float *Voltage, float *Power,
       float *Resistance, uint8_t *CurrentRange, uint8_t *ChMode);
```

A watchdog timer can be activated in the Misc_Options section of the System.ini file. Setting the MacNetDirectModeWD to a value higher than 0 will cause a channel under direct mode to suspend if it has not been updated within the specified number of seconds.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 8,
   "Chan": 3,
   "Current": 0.1,
   "Voltage": 20,
   "Power": 50,
   "Resistance": 0,
   "CurrentRange": 4,
   "ChMode": "C"
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":8,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 9) Set Variable

Set one of the 15 VAR's of channel given in Chan.

Transmit message:

Function Class	WORD	6
Function No	WORD	9
Chan	WORD	Test channel. 0-based.
Len	WORD	5
VarNum	Byte	Variable number. 1..15
VARiable	Single	Variable value

MacNetLib function prototype:

```
int32_t PingMacNet_6_9(uint16_t *Chan, uint8_t *VarNum, float *VARiable);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 9,
   "Chan": 3,
   "VarNum": 3
   "Value": -1.25
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":9,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```


(6, 10) Set safety

Set the safety of the channel given in Chan. See Channel Specs for more info.

Transmit message:

Function Class	WORD	6
Function No	WORD	10
Chan	WORD	Test channel. 0-based.
Len	WORD	5
VSafeMax	Single	
VSafeMin	Single	
ISafeChg	Single	
ISafeDis	Single	
PBatSafeChg	Single	
PBatSafeDis	Single	

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
```

```
   "FClass": 6,
   "FNum": 10,
   "Chan": 3,
   "VSafeMax": 5.0
   "VSafeMin": 0.1
   "ISafeChg": 4.3
   "ISafeDis": 2.5
   "PBatSafeChg": 0
   "PBatSafeDis": 0
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":10,
   "Chan":3,
   "VSafeMax":5,
   "VSafeMin":0.100000001490116,
   "ISafeChg":4.30000019073486,
   "ISafeDis":2.5,
   "PBatSafeChg":0,
   "PBatSafeDis":0
 }
 ,
 "id":1987
}
```


(6, 11) Check test start on channel

Checks if a test can be started with these start data in the “Data” area.

Transmit message:

Function Class	WORD	6
Function No	WORD	11
Chan	WORD	Test channel
Len	WORD	137
StartDataType	Byte	1. Reserved for future use
StartDataVersion	Byte	1. Reserved for future use
TestName	25 bytes	Up to 25 ascii characters for the data file name. “Random” will generate a pseudo-random name. The field is fixed in length, so remaining characters should be space characters.
ProcName	25 bytes	Up to 25 ascii characters for the test procedure name. The test procedure must exist in the C:\Maccor\Procedur folder and the “.000” should not be part of the name. The field is fixed in length, so remaining characters should be space characters.
Comment	80 bytes	Up to 80 characters for the test comment. The field is fixed in length, so remaining characters should be space characters.
Crate	Single	C-rate. If C-rate is not used, enter 1
ChamberNum	Byte	Environmental chamber number. If the environmental chamber is not used, enter 0.

A number will be returned describing the success or failure of the start:

Result	Word	0: OK 1: Channel not available or selected 2: Procedure does not exist 3: Sub routine procedures do not exist 4: File name exists in Archive 5: Invalid file name 6: Invalid EV chamber number 7: Compile error. Use function (6, 12) for details 0xFFFF: Other non-defined problem

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 11,
   "Chan": 3,
   "TestName": "Random",
   "ProcName": "Procedure Name",
   "Comment": "Test comment",
   "Crate": 1,
   "ChamberNum": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":6,
   "FNum":11,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(6, 12) Detailed check start error message

Get error messages reported by (6, 11) Check test start on channel.

Transmit message:

Function Class	WORD	6
Function No	WORD	12
Chan	WORD	-

Len	WORD	0

Reply message:

Function Class	WORD	6
Function No	WORD	12
Chan	WORD	-
Len	WORD	
Error message	Len number of bytes	Error message as ascii characters

NOTE: Keep calling this function until no string is returned to clear the error message buffer.

JSON

The functionality of this function is covered by (6, 11).

(6, 13) Advanced start

Starts or re-starts a test channel with advanced features. For further details of the start data please see Advanced Start.

Transmit message:

Function Class	WORD	6
Function No	WORD	13
Chan	WORD	Test channel to start. The channel must be available or selected. If other channels are selected, they will be dis-selected
Len	WORD	199
StartDataType	Byte	1. Reserved for future use
StartDataVersion	Byte	1. Reserved for future use
TestName	25 bytes	Up to 25 ascii characters for the data file name. “Random” will generate a pseudo-random name. The field is fixed in length, so remaining characters should be space characters. If an archived test is re-started, this field is not used.
ProcName	25 bytes	Up to 25 ascii characters for the test procedure name. The test procedure must exist in the C:\Maccor\Procedur folder and the “.000” should not be part of the name. The field is fixed in length, so remaining characters should be space characters. If an archived test is re-started and the procedure from the data file is used, this field is not used.
Comment	80 bytes	Up to 80 characters for the test comment. The field is fixed in length, so remaining characters should be space characters. If an archived test is re-started, this field is not used.
Crate	Single	C-rate. If C-rate is not used, enter 1
ChamberNum	Byte	Environmental chamber number. If the environmental chamber is not used, enter 0.
Options	Byte bitmap	Bit0: Append to Archived test Bit1: Use procedure from archived test. Bit0 must be set Bit2: Add shelf time. Bit0 must be set
Archived test name	30 bytes	Up to 30 ascii characters for the archived data file name including the channel number extension. Only used when Option.B0 is set.
Remove last data records	Byte	Number of data records to remove from the end of the file to be restarted. Only used when Option.B0 is set. Max 20 records and only back to the last header.
Test time	Single	In seconds. Enter -1 to use default start value or value from restart file
Step time	Single	In seconds. Enter -1 to use default start value or value from restart file
Test step	Int16	Enter -1 to use default start value or value from restart file

Cycle	Int32	Enter -1 to use default start value or value from restart file
Remaining Loop 1	Int16	Enter -1 to use default start value or value from restart file
Remaining Loop 2	Int16	Enter -1 to use default start value or value from restart file
Remaining Loop 3	Int16	Enter -1 to use default start value or value from restart file
Remaining Loop 4	Int16	Enter -1 to use default start value or value from restart file
Capacity	Single	In Ah. Enter -1 to use default start value or value from restart file
Energy	Single	In Wh. Enter -1 to use default start value or value from restart file

A number will be returned describing the success or failure of the start:

Result	Word	0: OK 1: Invalid channel number 2: Channel not available or selected 3: Illegal combination of options 4: Archived file does not exist 5: Procedure from archived file does not exist 6: Illegal test step 7: Cannot start at a blank step type 8: Cannot start at this step type 9: Cannot remove the requested number of data points 0xFFFF: Other non-defined problem

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 6,
   "FNum": 13,
   "Chan": 3,
   "TestName": "Random",
   "ProcName": "Procedure Name",
   "Comment": "Test comment",
   "Crate": 1,
   "ChamberNum": 0,
   "Options": 0,
   "ArchivedTestName": "Archived File.003",
   "RemoveLastDataRecords": 0,
   "TestTime": 0,
   "StepTime": 0,
   "TestStep": 1,
   "Cycle": 0,
   "RemainingLoop1": 0,
   "RemainingLoop2": 0,
   "RemainingLoop3": 0,
   "RemainingLoop4": 0,
   "Capacity": 0,
   "Energy": 0
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
```

```
 "result":
 {
   "FClass":6,
   "FNum":13,
   "Chan":3,
   "Result":"OK"
 }
 ,
 "id":1987
}
```


Function Class 7: Smart Battery

Enter topic text here.

(7, 1) Version and status info

Request Basic information about the smart battery channel assigned to “Chan” channel.

Transmit message:

Function Class	WORD	7
Function No	WORD	1
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	7
Function No	WORD	1
Chan	WORD	
Len	WORD	
SB Board num	Byte	
SB Position num	Byte	
SB Type	Byte	Can be 1, 2 or 3 for SMB1, SMB2, SMB3. SMB2 is obsolete and should never appear
Server version Major	Byte	SMB3 only
Server version Minor	Byte	SMB3 only
Client version Major	Byte	SMB3 only
Client version Minor	Byte	SMB3 only
DLL Version Major	Byte	As specified in the customer written DLL
DLL Version Minor	Byte	As specified in the customer written DLL
DLL Compile Num	Word	As specified in the customer written DLL
DLL version String	50 bytes	Up to 50 characters as specified in the customer written DLL
DLL Name	50 bytes	Name of the customer written DLL. Up to 50 characters.
SecsLastRead	Dword	Milliseconds since last successful read of entire scan list. 0 if never read.
SecsLastHeader	Dword	Milliseconds since last successful read of entire header. 0 if never read.
Status	Byte	0: Problem 1: Client idle 2: Reading 3: Reading 4: Scanlist downloaded 5: Scanlist DL timeout 6: Boot loader running

		128: Verifying client 129: Verify OK 130: Verify FAIL 131: Programming client 132: Programming done 133: Programming abort 134: Programming timeout
Bus voltage	Word	If supported. The bus logic voltage in mV.
Temperature	Int16 * 10	If supported. Temperature in Celsius times 10.

MacNetLib function prototype:

```
int32_t PingMacNet_7_1(uint16_t *Chan, uint8_t *SBbrd, uint8_t *SBPos, uint8_t
*SBType,
       uint8_t *ServerVersionMajor, uint8_t *ServerVersionMinor,
       uint8_t *ClientVersionMajor, uint8_t *ClientVersionMinor,
       uint8_t *DLLVersionMajor, uint8_t *DLLVersionMinor,
       uint16_t *DLLCompileNum, CStr DLLversionString, CStr DLLName,
       uint32_t *SecsLastRead, uint32_t *SecsLastHeader, uint8_t *Status, uint16_t
*BusVolt, int16_t *Temperature);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 1,
   "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":1,
   "Chan":9,
   "SBbrd":0,
   "SBPos":9,
   "SBType":3,
   "ServerVersionMajor":2,
   "ServerVersionMinor":1,
   "ClientVersionMajor":2,
   "ClientVersionMinor":7,
   "DLLVersionMajor":0,
   "DLLVersionMinor":0,
   "DLLCompileNum":0,
   "SecsLastRead":0.031,
   "SecsLastHeader":6.703,
   "Status":"Reading",
   "BusVolt":3.299,
```

```
   "Temperature":-48.4
 }
 ,
 "id":1987
}
```


Please note that SecsLastRead and SecsLastHeader are in seconds, BusVolt in volt and Temperature in Celsius. The is different from the binary version.

(7, 2) Suspend smart battery polling

Suspend smart battery polling of the battery assigned to “Chan”.

Transmit message:

Function Class	WORD	7
Function No	WORD	2
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 2,
   "Chan": 9
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":2,
   "Chan":9,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(7, 3) Resume smart battery polling

Resume smart battery polling of the battery assigned to “Chan”.

Transmit message:

Function Class	WORD	7
Function No	WORD	3
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 3,
   "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":3,
   "Chan":9,
   "Result":"OK"
 }
 ,
 "id":1987
}
```

(7, 4) Get data from a register

Request a word of data from a specific register of the battery assigned to “Chan”. The register must be setup in the scan list, and the value returned is from the last successful poll. The address of the register to read from is the first byte in Data. If the requested register is in the scan list, the register address will be returned in the first byte in Data and the data word in the two following bytes. If the register is not in the scan list, all three first bytes of data will be 0.

Transmit message:

Function Class	WORD	7
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	1

SMB register address	Byte	The register must be setup in the scan list

or

Function Class	WORD	7
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	2
Device address	Byte	The device must be setup in the scan list
Register address	Byte	The register must be setup in the scan list

Reply message:

Function Class	WORD	7
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	
SMB register address	Byte	If the requested register is in the scan list, the register address will be returned. If the register is not in the scan list, 0 will be returned
SMB data	WORD	The value returned is from the last successful poll.

or

Function Class	WORD	7
Function No	WORD	4
Chan	WORD	Test channel. 0-based.
Len	WORD	
Device address	Byte	If the requested device address is in the scan list, the register address will be returned. If the register is not in the scan list, 0 will be returned
Register address	Byte	If the requested register address is in the scan list, the register address will be returned. If the register is not in the scan list, 0 will be returned
SMB data	WORD	The value returned is from the last successful poll.

MacNetLib function prototype:

```
int32_t PingMacNet_7_4(uint16_t *Chan, uint8_t *RegAddr, uint16_t *Data);
```

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 4,
   "Chan": 9,
   "SMBDevAddr":22,
   "SMBRegAddr": 9
 },
 "id": 1987
}
```

Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":4,
   "Chan":9,
   "SMBDevAddr":22,
   "SMBRegAddr":9,
   "SMBRegValue":10261
 }
 ,
 "id":1987
}
```


(7, 5) Get the scan list

Return the scan list of the battery assigned to “Chan” as it is defined in the Smart Battery Setup in the tester program. The first byte is the number of registers and the subsequent three bytes for each register being: Address, Length and Type:

Transmit message:

Function Class	WORD	7
Function No	WORD	5
Chan	WORD	Test channel. 0-based.
Len	WORD	0: Addr, Ptr, Len replied. 1: DevAddr, RegAddr, Ptr, Len replied
Data	N/A	

Reply message if Len=0:

Function Class	WORD	7
Function No	WORD	5
Chan	WORD	
Len	WORD	
NumOfRegs	Byte	Number of registers to be polled as defined in MacTest32
Addr01	Byte	Address of first register
Ptr01	Byte	Pointer to the first register
Len01	Byte	Length of the first register
Addr02	Byte	Address of second register
Ptr02	Byte	Pointer to the second register
Type02	Byte	Length of the second register:
...		
...		
...		

Reply message if Len=1:

Function Class	WORD	7
Function No	WORD	5
Chan	WORD	
Len	WORD	

NumOfRegs	Byte	Number of registers to be polled as defined in MacTest32
DevAddr01	Byte	Device Address of first register
RegAddr01	Byte	Register Address of first register
Ptr01	Byte	Pointer to the first register
Len01	Byte	Length of the first register
DevAddr02	Byte	Device Address of second register
RegAddr02	Byte	Register Address of second register
Ptr02	Byte	Pointer to the second register
Type02	Byte	Length of the second register:
...		
...		
...		
...		

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 5,
   "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":5,
   "Chan":9,
   "NumOfRegs":32,
   "ScanList":
   [
     {
       "DevAddr":22,
       "Addr":1,
       "Ptr":0,
       "Len":2
     }
     ,
     {
       "DevAddr":22,
       "Addr":2,
       "Ptr":2,
       "Len":2
     }
     ,
     {
       "DevAddr":22,
```

```
  "Addr":3,
  "Ptr":4,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":4,
  "Ptr":6,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":5,
  "Ptr":8,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":4,
  "Ptr":10,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":6,
  "Ptr":12,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":4,
  "Ptr":14,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":7,
  "Ptr":16,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":8,
  "Ptr":18,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":9,
  "Ptr":20,
  "Len":2
}
,
```

```
{
  "DevAddr":22,
  "Addr":10,
  "Ptr":22,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":11,
  "Ptr":24,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":12,
  "Ptr":26,
  "Len":2
}
,
{
  "Addr":13,
  "Ptr":28,
  "Len":2
}
,
{
  "Addr":14,
  "Ptr":30,
  "Len":2
}
,
{
  "Addr":15,
  "Ptr":32,
  "Len":2
}
,
{
  "Addr":16,
  "Ptr":34,
  "Len":2
}
,
{
  "Addr":17,
  "Ptr":36,
  "Len":2
}
,
{
  "Addr":18,
  "Ptr":38,
  "Len":2
}
,
{
  "Addr":19,
  "Ptr":40,
  "Len":2
```

```
}
,
{
  "Addr":20,
  "Ptr":42,
  "Len":2
}
,
{
  "Addr":21,
  "Ptr":44,
  "Len":2
}
,
{
  "Addr":22,
  "Ptr":46,
  "Len":2
}
,
{
  "Addr":23,
  "Ptr":48,
  "Len":2
}
,
{
  "Addr":0,
  "Ptr":50,
  "Len":2
}
,
{
  "Addr":47,
  "Ptr":52,
  "Len":2
}
,
{
  "Addr":47,
  "Ptr":54,
  "Len":2
}
,
{
  "Addr":60,
  "Ptr":56,
  "Len":2
}
,
{
  "Addr":61,
  "Ptr":58,
  "Len":2
}
,
{
  "Addr":62,
  "Ptr":60,
  "Len":2
}
```

```
    ,
    {
      "Addr":63,
      "Ptr":62,
      "Len":2
    }
  ]
 }
 ,
 "id":1987
}
```

(7, 6) Get raw data

Return the Raw data of the battery assigned to “Chan”. Use the scan list obtained via Function (7, 5) to extract the data of the individual registers.

Transmit message:

Function Class	WORD	7
Function No	WORD	6
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	7
Function No	WORD	6
Chan	WORD	
Len	WORD	Number of bytes returned
Raw data	Len Bytes	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
FORMAT 1
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 6,
   "Chan": 9,
   "Format": 1
 },
 "id": 1987
}

FORMAT 2
{
```

```
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 6,
   "Chan": 9,
   "Format": 2
 },
 "id": 1987
}
```


Reply message (Example):

FORMAT 1


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":6,
   "Chan":9,
   "Format":1,
   "Len":64,
   "RawData":
   [
     100,
     0,
     10,
     0,
     1,
     96,
     0,
     0,
     255,
     255,
     0,
     0,
     255,
     255,
     0,
     0,
     1,
     0,
     153,
     11,
     21,
     40,
     0,
     0,
     0,
     0,
     1,
     0,
     0,
     0,
     0,
     0,
     0,
     0,
```

```
    193,
    25,
    255,
    255,
    255,
    255,
    255,
    255,
    240,
    17,
    250,
    50,
    208,
    2,
    2,
    0,
    0,
    0,
    20,
    0,
    20,
    0,
    0,
    0,
    105,
    13,
    77,
    13,
    95,
    13
  ]
 }
 ,
 "id":1987
}
```


FORMAT 2


```
{
 "jsonrpc":"2.0",
 "result":
 {
  "FClass":7,
  "FNum":6,
  "Chan":9,
  "Format":2,
  "Len":27,
  "Data":
  [
   {
    "Addr":1,
    "Len":2,
    "Data":
    [
     100,
     0
    ]
   }
   ,
   {
```

```
  "Addr":2,
  "Len":2,
  "Data":
  [
    10,
    0
  ]
}
,
{
  "Addr":3,
  "Len":2,
  "Data":
  [
    1,
    96
  ]
}
,
{
  "Addr":4,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":5,
  "Len":2,
  "Data":
  [
    255,
    255
  ]
}
,
{
  "Addr":4,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":6,
  "Len":2,
  "Data":
  [
    255,
    255
  ]
}
,
{
  "Addr":4,
```

```
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":7,
  "Len":2,
  "Data":
  [
    1,
    0
  ]
}
,
{
  "Addr":8,
  "Len":2,
  "Data":
  [
    154,
    11
  ]
}
,
{
  "Addr":9,
  "Len":2,
  "Data":
  [
    21,
    40
  ]
}
,
{
  "Addr":10,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":11,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":12,
  "Len":2,
```

```
  "Data":
  [
    1,
    0
  ]
}
,
{
  "Addr":13,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":14,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":15,
  "Len":2,
  "Data":
  [
    0,
    0
  ]
}
,
{
  "Addr":16,
  "Len":2,
  "Data":
  [
    193,
    25
  ]
}
,
{
  "Addr":17,
  "Len":2,
  "Data":
  [
    255,
    255
  ]
}
,
{
  "Addr":18,
  "Len":2,
  "Data":
```

```
    [
      255,
      255
    ]
  }
  ,
  {
    "Addr":19,
    "Len":2,
    "Data":
    [
      255,
      255
    ]
  }
  ,
  {
    "Addr":20,
    "Len":2,
    "Data":
    [
      240,
      17
    ]
  }
  ,
  {
    "Addr":21,
    "Len":2,
    "Data":
    [
      250,
      50
    ]
  }
  ,
  {
    "Addr":22,
    "Len":2,
    "Data":
    [
      208,
      2
    ]
  }
  ,
  {
    "Addr":23,
    "Len":2,
    "Data":
    [
      2,
      0
    ]
  }
  ,
  {
    "Addr":0,
    "Len":2,
    "Data":
    [
```

```
        0,
        0
      ]
    }
    ,
    {
      "Addr":47,
      "Len":2,
      "Data":
      [
        20,
        0
      ]
    }
  ]
 }
 ,
 "id":1987
}
```


(7, 7) Get the header scan list

Return the header scan list of the battery assigned to “Chan” as it is defined in the Smart Battery Setup in the tester program. The first byte is the number of registers and the subsequent three bytes for each register being: Address, Length and Type:

Transmit message:

Function Class	WORD	7
Function No	WORD	7
Chan	WORD	Test channel. 0-based.
Len	WORD	0: Addr, Ptr, Len replied. 1: DevAddr, RegAddr, Ptr, Len replied
Data	N/A	

Reply message if Len=0:

Function Class	WORD	7
Function No	WORD	7
Chan	WORD	
Len	WORD	
NumOfRegs	Byte	Number of registers to be polled as defined in MacTest32
Addr01	Byte	Address of first register
Ptr01	Byte	Pointer to the first register
Len01	Byte	Length of first register
Addr02	Byte	Address of second register
Ptr02	Byte	Pointer to the second register
Len02	Byte	Length of second register
...		
...		
...		

Reply message if Len=1:

Function Class	WORD	7
Function No	WORD	7
Chan	WORD	
Len	WORD	

NumOfRegs	Byte	Number of registers to be polled as defined in MacTest32
DevAddr01	Byte	Device Address of first register
RegAddr01	Byte	Register Address of first register
Ptr01	Byte	Pointer to the first register
Len01	Byte	Length of the first register
DevAddr02	Byte	Device Address of second register
RegAddr02	Byte	Register Address of second register
Ptr02	Byte	Pointer to the second register
Type02	Byte	Length of the second register:
...		
...		
...		
...		

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 7,
   "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":7,
   "Chan":9,
   "NumOfRegs":12,
   "ScanList":
   [
     {
       "DevAddr":22,
       "Addr":0,
       "Ptr":0,
       "Len":2
     }
     ,
     {
       "DevAddr":22,
       "Addr":1,
       "Ptr":2,
       "Len":2
     }
     ,
     {
```

```
  "DevAddr":22,
  "Addr":2,
  "Ptr":4,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":24,
  "Ptr":6,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":25,
  "Ptr":8,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":26,
  "Ptr":10,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":27,
  "Ptr":12,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":28,
  "Ptr":14,
  "Len":2
}
,
{
  "DevAddr":22,
  "Addr":32,
  "Ptr":16,
  "Len":8
}
,
{
  "DevAddr":22,
  "Addr":33,
  "Ptr":24,
  "Len":4
}
,
{
  "DevAddr":22,
  "Addr":34,
  "Ptr":28,
  "Len":4
}
```

```
    ,
    {
      "DevAddr":22,
      "Addr":35,
      "Ptr":32,
      "Len":25
    }
  ]
 }
 ,
 "id":1987
}
```

(7, 8) Get raw header data

Return the Raw header data of the battery assigned to “Chan”. Use the scan list obtained via Function (7, 7) to extract the data of the individual registers.

Transmit message:

Function Class	WORD	7
Function No	WORD	8
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	7
Function No	WORD	8
Chan	WORD	
Len	WORD	Number of bytes returned
Raw data	Len Bytes	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

FORMAT 1

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 8,
   "Chan": 9,
   "Format": 1
 },
 "id": 1987
}
```

FORMAT 2

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 7,
   "FNum": 8,
   "Chan": 9,
   "Format": 2
 },
 "id": 1987
}
```


Reply message (Example):

FORMAT 1


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":8,
   "Chan":9,
   "Format":1,
   "Len":57,
   "RawData":
   [
     0,
     0,
     100,
     0,
     10,
     0,
     159,
     25,
     156,
     44,
     49,
     0,
     141,
     69,
     1,
     0,
     83,
     77,
     80,
     153,
     153,
     153,
     153,
     153,
     98,
     113,
     50,
     48,
     76,
     73,
     79,
```

```
    78,
    0,
    0,
    0,
    0,
    7,
    2,
    0,
    1,
    50,
    23,
    0,
    0,
    3,
    51,
    48,
    56,
    3,
    48,
    48,
    55,
    3,
    83,
    68,
    73,
    2
   ]
 }
 ,
 "id":1987
}
```


FORMAT 2


```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":7,
   "FNum":8,
   "Chan":9,
   "Format":2,
   "Len":12,
   "Data":
   [
     {
       "Addr":0,
       "Len":2,
       "Data":
       [
         0,
         0
       ]
     }
     ,
     {
       "Addr":1,
       "Len":2,
       "Data":
       [
         100,
```

```
    0
  ]
}
,
{
  "Addr":2,
  "Len":2,
  "Data":
  [
    10,
    0
  ]
}
,
{
  "Addr":24,
  "Len":2,
  "Data":
  [
    159,
    25
  ]
}
,
{
  "Addr":25,
  "Len":2,
  "Data":
  [
    156,
    44
  ]
}
,
{
  "Addr":26,
  "Len":2,
  "Data":
  [
    49,
    0
  ]
}
,
{
  "Addr":27,
  "Len":2,
  "Data":
  [
    141,
    69
  ]
}
,
{
  "Addr":28,
  "Len":2,
  "Data":
  [
    1,
    0
```

```
  ]
 }
 ,
 {
  "Addr":32,
  "Len":8,
  "Data":
  [
    83,
    77,
    80,
    153,
    153,
    153,
    153,
    153
  ]
 }
 ,
 {
  "Addr":33,
  "Len":4,
  "Data":
  [
    98,
    113,
    50,
    48
  ]
 }
 ,
 {
  "Addr":34,
  "Len":4,
  "Data":
  [
    76,
    73,
    79,
    78
  ]
 }
 ,
 {
  "Addr":35,
  "Len":25,
  "Data":
  [
    0,
    0,
    0,
    0,
    7,
    2,
    0,
    1,
    50,
    23,
    0,
    0,
    3,
```

```
        51,
        48,
        56,
        3,
        48,
        48,
        55,
        3,
        83,
        68,
        73,
        2
      ]
    }
  ]
 }
 ,
 "id":1987
}
```

(7, 9) Write multiple bytes A

Write up to 60 bytes of data to a specific register of the battery assigned to “Chan”. The address of the register to write to is the first byte in Data, the number of bytes to write the second and the data bytes to write in the following bytes. The reply provides no information of the success of the write due to the time it may take and a subsequent read should be performed for verification.

Transmit message:

Function Class	WORD	7
Function No	WORD	9
Chan	WORD	Test channel. 0-based.
Len	WORD	DataLen + 2
RegAddr	Byte	Address of register to write to
DataLen	Byte	Number of bytes to write. Max. 60
Data	DataLen Bytes	Data to write

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 9,
 "Chan": 9,
 "RegAddr": 1,
 "Data": [44, 1]
 },
 "id": 1987
}
```

Note that the "DataLen" is the length of the "Data" array

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":9,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```

(7, 10) Write multiple bytes B

Write up to 256 bytes of data to a specific register or a consecutive range of registers of the battery assigned to “Chan”.

The reply provides no information of the success of the write due to the time it may take and a subsequent read should be performed for verification.

Transmit message:

Function Class	WORD	7
Function No	WORD	10
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of bytes
BusAddr	Byte	0x16: I^2C and SMB
StartDataAddr	Byte	First address of register to write to
EndDataAddr	Byte	Last address of register to write to. If StartDataAddr = EndDataAddr all data are written to the same address. If StartDataAddr < EndDataAddr the data are written one byte to each address in the range.
NumOfBytes	Byte	Number of bytes to write to a single address
BusType	Byte	01: I^2C and SMB 02: HDQ 03: Future 1-wire
CheckSum	Byte	Checksum for the actual device
Data	Bytes	Data to write

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 10,
```

```
 "Chan": 9,
 "BusAddr": 22,
 "StartDataAddr": 1,
 "EndDataAddr": 1,
 "NumOfBytes": 2,
 "BusType": 1,
 "CheckSum": 0,
 "Data": [44, 1]
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":10,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 11) Request read multiple bytes

Read up to 256 bytes of data from a specific register or a consecutive range of registers of the battery assigned to “Chan”. It may take up to 2 seconds for the requested data to be uploaded from the device. The data are cached and can subsequently be read with Function No 12.

The reply provides no information of the success of the read due to the time it may take and a subsequent read should be performed for verification.

Transmit message:

Function Class	WORD	7
Function No	WORD	11
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of bytes
BusAddr	Byte	0x16: I^2C and SMB
StartDataAddr	Byte	First address of register to read from
EndDataAddr	Byte	Last address of register to read from. If StartDataAddr = EndDataAddr all data are read from the same address. If StartDataAddr < EndDataAddr the data are read one word from each address in the range.
NumOfBytes	Byte	Number of bytes to read from a single address
BusType	Byte	01: I^2C and SMB 02: HDQ 03: Future 1-wire

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 11,
 "Chan": 9,
 "BusAddr": 22,
 "StartDataAddr": 34,
 "EndDataAddr": 34,
 "NumOfBytes": 4,
 "BusType": 1
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":11,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 12) Return the bytes read

Read the data requested with Function No (7, 11) of the battery assigned to “Chan”.

Transmit message:

Function Class	WORD	7
Function No	WORD	12
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Reply message:

Function Class	WORD	7
Function No	WORD	12
Chan	WORD	Test channel. 0-based.
Len	WORD	Number of bytes
Status	Byte	1: Read in progress 2: Read completed 3: Error
NumOfBytes	Word	Number of bytes requested
Data	Bytes	Data from device

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 12,
 "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":12,
 "Chan":9,
 "Status":2,
 "NumOfBytes":4,
 "Data":
 [
   76,
   73,
   79,
   78
 ]
 },
 "id": 1987
}
```

(7, 13) Read header from battery

Request the smart battery header of the battery assigned to “Chan” to be read from the battery. The test channel must be in an available state. This takes several seconds to complete.

When the header has been read from the battery, the data are available by calls to (7, 7) and (7, 8)

Transmit message:

Function Class	WORD	7
Function No	WORD	13
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 13,
 "Chan": 9
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":13,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 14) Get position setup

Return the position setup list of the battery assigned to “Chan” as it is set in the Smart Battery Setup in the tester program.

Transmit message:

Function Class	Word	7
Function No	Word	14
Chan	Word	Test channel. 0-based.
Len	Word	0
Data	N/A	

Reply message:

Function Class	Word	7
Function No	Word	14
Chan	Word	
Len	Word	
BusType	Byte	1: SMB and I2C 2: HDQ8 3: HDQ16 4: 1-Wire®
DeviceAddr	Byte	Device address
ClockFreqDivisor	Word	Clock frequency divisor
BusLogicV	Single	Bus logic voltage in volts
AccessDelay	Word	Access delay in ms
PctOfMax	Byte	Scan list size in percent of max possible size
PctOfMaxHdr	Byte	Header scan list size in percent of max possible size

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 14,
 "Chan": 9
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":14,
 "Chan":9,
 "BusType":1,
 "DeviceAddr":22,
 "ClockFreqDivisor":85,
 "BusLogicV":3.29934072494507,
 "AccessDelay":2,
 "PctOfMax":56,
 "PctOfMaxHdr":14
 },
 "id": 1987
}
```

(7, 15) Set position setup

Set the position setup list of the battery assigned to “Chan” as it is set in the Smart Battery Setup in the tester program. Only possible when the channel is available.

NOTE!

Not all values and combinations of values are possible. Check carefully what is required for the actual device.

Transmit message:

Function Class	Word	7
Function No	Word	15
Chan	Word	Test channel. 0-based.
Len	Word	12
BusType	Byte	1: SMB and I2C 2: HDQ8 3: HDQ16 4: 1-Wire®
DeviceAddr	Byte	Device address
ClockFreqDivisor	Word	Clock frequency divisor

BusLogicV	Single	Bus logic voltage in volts
AccessDelay	Word	Access delay in ms
PctOfMax	Byte	N/A
PctOfMaxHdr	Byte	N/A

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 15,
 "Chan": 9,
 "BusType":1,
 "DeviceAddr":22,
 "ClockFreqDivisor":85,
 "BusLogicV":3.3,
 "AccessDelay":2
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":15,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```

(7, 20) Read one entry of scan list

Read one entry of the software version of the scan list of the battery assigned to “Chan”. The index of the scan list entry is the first data byte.

Transmit message:

Function Class	WORD	7
Function No	WORD	20
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Index	Byte	Scan list index

Reply message:

Function Class	WORD	7

Function No	WORD	20
Chan	WORD	Test channel. 0-based.
Len	WORD	
Index	Byte	Scan list index
RegAddr	Byte	Register address
Len	Byte	Register length
ErrorVal	Word	Value to use if a read error is encountered
DataType	Byte	0: Byte 1: Word 2: String 3: Word+ 4: Word-
Designator	25 bytes	Up to 25 characters for the designator.
FormatType	Word	0: Hex 1: Binary 2: Unsigned 3: Signed 4: StringASCII 5: StringHEX 6: Float2SD10 7: Float2SD100 8: Float2SD1000 9: Float2UD10 10: Float2UD100 11: Float2UD1000 12: Celsius 13: Date
DevAddr	Byte	Device address

Please refer to the general manual and the SMB configuration screen for further details.

Currently not supported in MacNetLib.

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 20,
 "Chan": 9
 "Index": 3
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":20,
 "Chan":9,
 "Index":3,
```

```
 "dev_addrs":22,
 "reg_addrs":4,
 "Len":2,
 "data_type":2,
 "ErrorVal":65534,
 "Designator":"AtRate",
 "FormatType":2
 },
 "id": 1987
}
```


(7, 21) Write one entry of scan list

Write the one entry of the software version of the scan list of the battery assigned to “Chan”. The channel must be in an available state.

Transmit message:

Function Class	WORD	7
Function No	WORD	21
Chan	WORD	Test channel. 0-based.
Len	WORD	33
Index	Byte	Scan list index
RegAddr	Byte	Register address
Len	Byte	Register length
ErrorVal	Word	Value to use if a read error is encountered
DataType	Byte	0: Byte 1: Word 2: String 3: Word+ 4: Word-
Designator	25 bytes	Up to 25 characters for the designator.
FormatType	Word	0: Hex 1: Binary 2: Unsigned 3: Signed 4: StringASCII 5: StringHEX 6: Float2SD10 7: Float2SD100 8: Float2SD1000 9: Float2UD10 10: Float2UD100 11: Float2UD1000 12: Celsius 13: Date
DevAddr	Byte	Device address

Please refer to the general manual and the SMB configuration screen for further details.

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
```

```
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass": 7,
 "FNum": 21,
 "Chan": 9,
 "Index": 3,
 "dev_addrs":22,
 "reg_addrs": 4,
 "Len": 2,
 "data_type": 2,
 "ErrorVal": 65534,
 "Designator": "AtRateX",
 "FormatType": 2
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":21,
 "Chan":9,
 "Index":3,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 22) Clear the scan list

Clear the software version of the scan list of the battery assigned to “Chan”. The channel must be in an available state.

Transmit message:

Function Class	WORD	7
Function No	WORD	22
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
```

```
 "params":
 {
 "FClass": 7,
 "FNum": 22,
 "Chan": 9
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
 "FClass":7,
 "FNum":22,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 23) Save the updated scan list

Save the updated software version of the scan list of the battery assigned to “Chan” and update the firmware scan list. The channel must be in an available state. Since this involves an update in the firmware the reply time will be at least 300 ms.

Transmit message:

Function Class	WORD	7
Function No	WORD	23
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
 "FClass": 7,
 "FNum": 23,
 "Chan": 9
 },
 "id": 1987
}
```


Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
 "FClass":7,
 "FNum":23,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```


(7, 24) Refresh scan list

Refresh firmware scan list of the battery assigned to “Chan”. Since this involves an update in the firmware the reply time will be at least 300 ms.

Transmit message:

Function Class	WORD	7
Function No	WORD	24
Chan	WORD	Test channel. 0-based.
Len	WORD	0
Data	N/A	

Currently not supported in MacNetLib.

JSON

Transmit message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
 "FClass": 7,
 "FNum": 24,
 "Chan": 9
 },
 "id": 1987
}
```


Reply message (Example):


```
{
 "jsonrpc":"2.0",
 "result":
 {
 "FClass":7,
 "FNum":24,
 "Chan":9,
 "Result":"OK"
 },
 "id": 1987
}
```

Function Class 8: Authentication

Enter topic text here.

(8, 1) Log in

If user authentication is activated, the user must log in with the given user ID and password. Note: The password is case sensitive.

Transmit message:

Function Class	WORD	8
Function No	WORD	1
Chan	WORD	
Len	WORD	2 + UserIDLen + PasswordLen
UserIDLen	Byte	Length of user ID
UserID	UserIDLen bytes	User ID ascii characters
PasswordLen	Byte	Length of Password
Password	PasswordLen bytes	Password ascii characters

Reply message:

Function Class	WORD	8
Function No	WORD	1
Chan	WORD	
Len	WORD	1
Result	Byte	1: Authentication passed; 0: Authentication failed

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 8,
   "FNum": 1,
   "UserID": "Robin Hood",
   "Password": "xxxxyyyy"
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":8,
   "FNum":1,
   "Result":"PASSED"
 }
 ,
 "id":1987
}
```

(8, 2) Log out

If user authentication is activated, this function will log out a logged in user.

Transmit message:

Function Class	WORD	8
Function No	WORD	2
Chan	WORD	
Len	WORD	0

Reply message:

Function Class	WORD	8
Function No	WORD	2
Chan	WORD	
Len	WORD	1
Result	Byte	1: Log out success; 0: Log out failed

JSON

Transmit message (Example):

```
{
 "jsonrpc": "2.0",
 "method": "MacNet",
 "params":
 {
   "FClass": 8,
   "FNum": 2
 },
 "id": 1987
}
```

Reply message (Example):

```
{
 "jsonrpc":"2.0",
 "result":
 {
   "FClass":8,
   "FNum":2,
   "Result":"LOGGED OUT"
 }
 ,
 "id":1987
}
```

File based remote control

Obsolete Feature: Please contact Maccor Customer Service for instructions in using MacNet.