PCYP User Manual

Model	CTM-200 R2
Revision	Rev 1.2

3066 Beta Avenue | Burnaby, B.C. | V5G 4K4

Revision Control

Description	Revision	Initials	Date
Initial version	1.0	cch	26 -Oct-15
Revised key and msg commands Removed value commands Added group command	1.1	cch	11-Jan-16
Added list of invalid delimiters	1.2	cch	14-Mar-16

Contents

1. Overview

This document defines the message structure of the PCYP message. The PCYP message moves away from the “hard coded” NMEA style messages, offering flexibility in message construction.

Below is an example of a typical device generated report containing a PGPS NMEA style message.

```
$PGPS,155708.00,A,4915.4268,N,12259.8155,W,002.6,141.0,070813,+00023,12,3562150400
95550*73[0x0D][0x0A]
```

The $PGPS (and all NMEA style) message is strongly typed, and is therefore pre-defined. This means modifying the message not possible, so adding or changing a field requires a new message type.

To allow generic user created messages, the PCYP message has been developed. The PCYP messages have the following benefits over of the NMEA style message:

- Individual parameter insertion
- User customization
- Server side flexibility
- Expandability

With PCYP messages, the values are stored as ASCII key/value pairs. The key/value pairs are added or removed as the user wishes. The server does not need any prior knowledge of a PCYP message contents, the key and value are simply read together as a pair. The full message content will be discussed in detail in the following sections.

2. PCYP Message

The message format is defined as follows:

```
$PCYP,flags=V,date=yymmdd,ts=hhmmss.00,id=<IMEI>,msgid=<ID>,key1=va
lue1,key2=value2,...,keyN=valueN,*hh<CR><LF>
```

The following fields are automatically appended into the message:

- $PCYP – message type
- flags – message flags
- date – date of the message
- ts – UTC time of the message
- id – modem ID
- msgid – message ID
- *hh – message checksum
- <CR><LF> - carriage return and linefeed

The key/value pairs are the user configurable entries into the PCYP message:

```
key1=value1,key2=value2,...,keyN=valueN
```

Some real examples that show the system version, WIFI type and RSSI are:

```
sys_ver=2.2.1.4900-91,wifi_type=IEEE802.11abgn,cell_rssi=-79
```

PCYP messages have no size limitation. Instead, as a message grows, it is automatically sub-divided into “chunks” of manageable data. Currently, the software divides PCYP messages into 255 byte chunks.

2.1 Delimiting

The PCYP message contains fields that are delimited by commas (,). Value strings that contain commas are searched and escape sequenced if necessary with a backslash (\).

The delimiting character is customizable by the user and can be changed to a character more suitable, however some characters are invalid:

- dollar sign: **$**
- equal sign: **=**
- backslash: ****
- asterisk: *****
- quotes, both single and double: **‘** and **“**
- alphanumerics: **0-9, A-Z, a-z**

Note, a value string that is escape sequenced may increase in size, however the same chunk size restriction is maintained by the software. Ie, escape characters are included in the size calculation when dividing into chunks.

2.2 Message Type

The message type follows the standard naming scheme for the existing CTM reporting.

The PCYP message always begins with the following:

```
$PCYP
```

2.3 Flags

Message flags are inserted into the message.

Currently, the GPS validity flag is inserted and follows the naming scheme: A,B,V,W as per existing reports.

Flag	Description
A	Valid current data
B	Valid stored data
V	Invalid current data
W	Invalid stored data

2.4 Date

The message date is inserted into the message with the format:
YYMMDD

2.5 Timestamp

The UTC timestamp is inserted into the message with the format:
HHMMSS.00

2.6 Modem ID

The modem ID uniquely identifies the unit to the server and is typically the IMEI.

2.7 Message ID

The message ID is a mandatory field that is automatically populated into the message. This field allows distinction between other PCYP messages. Valid values are 1000-1999.

2.8 Key/Value Pairs

A key/value pair consists of a *key*, which tells the parsing application what the parameter is, and the *value*, which is parameter's value. Key/value pairs are delimited by an equals sign (=) without any spaces. This delimiter cannot be changed. If a value string contains equals sign characters, they are escape sequenced by a backslash (\).Value strings that contain backslashes are also escaped with a backslash.

Multiple key/value pair entries are contained within a PCYP message up to the maximum number of entries, N. This is constrained by the maximum size of the entire message, 255 bytes.

A pre-defined list of keys is provided when constructing the PCYP message. In addition, the user has the capability of creating user-defined keys and setting their corresponding value.

2.9 Checksum

The final field in the PCYP message is the checksum. The format for this field is:

`*hh<CR><LF>`

The format is identical to the existing reporting scheme.

Parameter	Description
*	ASCII asterisk
hh	2-byte checksum (XOR all data bytes)
CR	Carriage return
LF	Line feed

3. User Interface

The PCYP message user interface is via the command line cmds.

A user creates PCYP messages by following the procedure:

1. View the list of keys available to add to a PCYP message
2. Add the desired keys to the corresponding PCYP message
3. View or query the PCYP message to ensure the message is correct
4. Edit the PCYP message if necessary
5. Send the report

The commands in the following sections are shown in **bold** type. If the command has required parameters, they are also shown in bold, but also with angle brackets: **<parameter>**. Optional parameters are shown with square brackets: **[optional]**.

3.1 PCYP commands

The new top level command is called ***pcyp*** and contains sublevels. Some of the main sublevels are: ***key*** and ***msg***.

The commands in the following sections are shown in **bold** type. If the command has required parameters, they are also shown in bold, but also with angle brackets: **<parameter>**. Optional parameters are shown with square brackets: **[optional]**.

The **`cmd pcyp`** command shows a list of all the PCYP (sub)-commands available:

```
root@CTM200:/# cmd pcyp
cmd pcyp key
cmd pcyp group
cmd pcyp delimiter
cmd pcyp msg
```

As usual, the **show** command displays the output of all commands which is suitable for applying the same configuration to other units:

`cmd pcyp show`

```
root@CTM200:/# cmd pcyp show
cmd pcyp group add MY_GROUP CELL
cmd pcyp delimiter ","
cmd pcyp msg add 1000 SYS_FW_LABEL
cmd pcyp msg add 1000 MY_GROUP
OK
```

3.2 Key commands

Keys are pre-defined and built into software. They are categorized as follows:

- Acceleration (ACCEL)
- Cellular Radio (CELL)
- GPS (GPS)
- OBD (OBD)
- Network Interface (NET)
- RFID (RFID)
- System (SYS)

3.2.1 Viewing the full list of keys

The purpose of this function is to allow the user to have a simple method to view keys with related information. This helps users to find names of keys when constructing messages.

To view all keys the following command is used:

```
cmd pcyp key view
```

Here is some sample output:

```
root@CTM200:/# cmd pcyp key view
Key                     Value                   Description
-------------------------------------------------------------------------------
CELL                                            named database for CELL RF data
CELL_RFQ                52                      overall rf quality in 0 - 100% (derived from indicators below)
CELL_RSSI               -68                     Receive Signal Strength Indicator (DC-HSPA: carrier 0)
.
.
.


SYS_FW_LABEL            2.3.0.5047-72           CTM fw version
SYS_FW_VER              2.3.0                   CTM fw version
SYS_FW_REV              5047                    CTM fw revision
SYS_STM_VER             72                      STM FW version
SYS_DEVICE_ID           359225050564319         configured CTM Gateway Device ID (cmd modemid)
SYS_DEVICE_ESN          359225050564319         CTM Gateway Device ESN/pseudo-ESN
SYS_TEMPERATURE         63                      system temperature
SYS_VCC                 11.740000               system supply voltage
SYS_DAY                 1075970055              day
SYS_MONTH               65537                   month
SYS_YEAR                16                      year
SYS_TIME                121403                  time
.
.
.


VEHICLE                                         named database
VEH_ENGHOURS            0                       system Engine Hours
VEH_IGN                 1                       ignition status
VEH_ODO                 42002                   system odometer value (from GPS/OBD/CAN)
VEH_ODO_SRC             0                       system odometer source
VEH_FUELUSED            0.000000                calculated Fuel Used value
VEH_FUELRATE            not found               calculate Fuel consumption rate (L/100Km X 100)
VEH_IDLETIME            0                       calculated Idle Time (in seconds)
SYS_ALARM_STATUS        1                       system io alarm status
OK
```

3.2.2 Viewing keys by wildcard

A user may wish to only display keys starting with a specific prefix:

cmd pcyp key view [prefix]

For example:

```
root@CTM200:/# cmd pcyp key view sys
SYS_FW_LABEL            2.3.0.5047-72           CTM fw version
SYS_FW_VER              2.3.0                   CTM fw version
SYS_FW_REV              5047                    CTM fw revision
SYS_STM_VER             72                      STM FW version
SYS_DEVICE_ID           359225050564319         configured CTM Gateway Device ID
(cmd modemid)
SYS_DEVICE_ESN          359225050564319         CTM Gateway Device ESN/pseudo-ESN
SYS_TEMPERATURE         63                      system temperature
SYS_VCC                 11.740000               system supply voltage
SYS_DAY                 1075970055              day
SYS_MONTH               65537                   month
SYS_YEAR                16                      year
SYS_TIME                121403                  time
OK
```

The above example outputs all keys that begin with the prefix “sys”. Note, the prefix is entered as lower case in the command, however is automatically matched to the upper case keys as stored in the database.

In order to output all keys starting with the letter “s”, the command is:

```
root@CTM200:/# cmd pcyp key view s
```

Conversely, in order to view only SYS_FW key entries, the command is:

```
root@CTM200:/# cmd pcyp key view sys_fw
```

3.3 Group commands

A group is simply a container that holds a number of pcyp keys, designed for simplifying message creation. Typically, keys are hand-picked and added to a group for a specific purpose. The CTM200 provides some pre-defined groups built into software, resembling the familiar NMEA style messages for OFIE, PGPS and PPWR.

For example, the PPWR group contains the keys:

- SYS_TEMPERATURE
- SYS_VCC
- SYS_ALARM_STATUS

3.3.1 Adding keys to a group

To create a new group, and add a key, the command is:

```
root@CTM200:/# cmd pcyp group add <group> <key>
```

For example:

```
root@CTM200:/# cmd pcyp group add MY_GROUP SYS_VER
OK
```

To add more keys to the same group, simply repeat the command with the same group name with the new keys:

```
root@CTM200:/# cmd pcyp group add MY_GROUP SYS_UPTIME
OK
```

MY_GROUP now contains SYS_VER and SYS_UPTIME.

3.3.2 Deleting Keys from a group

Keys are deleted from a group one at a time by the command:

cmd pcyp group del <group> [key]

Note: the key field is optional. If a key is not specified, the entire group will be deleted!

Also note, the special key value "all". All groups are deleted and cannot be recovered. Note however, the pre-defined groups are not affected by this command.

3.3.3 Replacing a key within a group

A key within a group can be replaced with a new key. This command is useful when a *typo* is found resulting in a key not found or a key is no longer needed and is to be replaced with a new key in the exact same ordered position.

A key is replace within a group by the command:

cmd pcyp group replace <group> <old key> <new key>

Example:

```
root@CTM200:/# cmd pcyp group replace MY_GROUP SYS_VCCC SYS_VCC
OK
```

3.3.4 Viewing groups

Groups can be viewed to determine the keys that are contained:

cmd pcyp group view [group]

If no group is specified, all groups are listed. If the group is specified and a group exists, only the contents of that group are displayed. The output of the command is of the form:

```
[GROUP] [KEY]
```

For example:

```
root@CTM200:/# cmd pcyp group view GR_PGPS
GR_PGPS GPS_TIME_OF_POS
GR_PGPS GPS_STATUS
GR_PGPS GPS_LAT_NMEA
GR_PGPS GPS_LAT_HEMI
GR_PGPS GPS_LON_NMEA
GR_PGPS GPS_LON_HEMI
GR_PGPS GPS_VEL_NMEA
GR_PGPS GPS_HDG_NMEA
GR_PGPS GPS_DAY
GR_PGPS GPS_MONTH
GR_PGPS GPS_YEAR
GR_PGPS GPS_ALT_NMEA
GR_PGPS GPS_SATS_INUSE
OK
```

In the example above, the group name is “GR_PGPS” while the keys are the “GPS_” keys.

Note, the group name must match identically to display the contents of a group.

3.4 Message commands

3.4.1 Constructing a PCYP message

Key/value pairs are added one at a time to a specific PCYP message. PCYP msg commands do not require the value since they are automatically matched to the key. To add a key/value pair, the syntax is:

cmd pcyp msg add <msgid> <key>

For example, to add key’s for the system version, “sys_ver” and system uptime, “sys_uptime” to message 1000:

```
root@CTM200:/# cmd pcyp msg add 1000 sys_ver
root@CTM200:/# cmd pcyp msg add 1000 sys_uptime
```

and “sys_ver” only to messages 1001 and 1002:

```
cmd pcyp msg add 1001 sys_ver
cmd pcyp msg add 1002 sys_ver
```

Results in creating 3 separate PCYP messages:

```
$PCYP,…,msgid=1000,sys_ver=2.2.1.4900-91,sys_uptime=128,*hh<CR><LF>

$PCYP,…,msgid=1001,sys_ver=2.2.1.4900-91,*ii<CR><LF>

$PCYP,…,msgid=1002,sys_ver=2.2.1.4900-91,*jj<CR><LF>
```

Keys can be added as lower case, but are automatically converted into upper case.

Note, keys are appended to the PCYP message in the order they are added to the command. Also, only 1 instance of a key can be added, ie, 2 of the same keys cannot be added to a single message.

The “msg add” command can be executed as many times as the user wishes, adding key/value pairs to the msg ID. As more keys are added to a specific message, PCYP messages are automatically subdivided into “chunks” in order to avoid truncation. This automatic subdivision occurs no matter how many keys or groups are added to a particular message. Because of this, if so desired, all required keys could be entered into a single monolithic pcyp message.

If a group is added to a message, all keys contained within the group are added to the pcyp message. In this case, duplicate keys can be added.

For example, if the same group as above is used, MY_GROUP, and is added to a message:

```
root@CTM200:/# cmd pcyp msg add 1001 my_group
```

The resulting pcyp message would be something like:

```
$PCYP,…,msgid=1001,sys_ver=2.2.1.4900-91,sys_uptime=128,*hh<CR><LF>
```

Note the group name is not present in the actual pcyp message. The group name is only a construct within the CTM200 pcyp engine. The end user simply parses the received message as a regular message.

3.4.2 Viewing the message keys

The following commands will dump the keys that were added to PCYP messages.

The view command provides the user an easy way to view the keys for a specific pcyp message:

cmd pcyp msg view [msgid]

For example:

```
root@CTM200:/# cmd pcyp msg view 1000
1000 sys_ver
1000 sys_uptime
```

Entering a specific message ID only displays the keys for that one message. All messages can also be viewed, without specifying a message ID:

```
root@CTM200:/# cmd pcyp msg view
1000 sys_ver
1000 sys_uptime
1001 sys_ver
1002 sys_ver
```

To display the full command with the parameters (to duplicate the configuration on another unit):

cmd pcyp msg [show]

```
root@CTM200:/# cmd pcyp msg show
cmd pcyp msg add 1000 "sys_ver"
cmd pcyp msg add 1000 "sys_uptime"
cmd pcyp msg add 1001 "sys_ver"
cmd pcyp msg add 1002 "sys_ver"
OK
```

3.4.3 Query a PCYP message as the actual $PCYP string

It may be useful to the user to have the capability to view the actual PCYP message string for verification before sending the report.

To view an individual message:
cmd pcyp msg query <msgid>

The output format of the query message is:

```
<msgid>:<length>:<pcyp message contents>
```

For example the following message 1000 contains system SQI and system uptime:

```
root@CTM200:/# cmd pcyp msg query 1000
1000:60:$PCYP,…,msgid=1000,sys_sqi=48,sys_uptime=738,*hh<CR><LF>
OK
```

3.4.4 Querying all PCYP messages

To view all pcyp messages, use the query command without any parameters.

```
cmd pcyp msg query
1000:60:$PCYP,…,msgid=1000,sys_sqi=48,sys_uptime=738,*hh<CR><LF>
1001:60:$PCYP,…,msgid=1001,cell_net=14,cell_rssi=-76,*ii<CR><LF>
.
.
.
N:60:$PCYP,…,msgid=N,key1=value1,key2=value2,*jj<CR><LF>
OK
```

3.4.5 Deleting a key from a message

An individual key can be deleted from a message:

cmd pcyp msg del <msgid> <key>

If a key is deleted, the corresponding key/value pair is removed from the message and the order of the remaining keys remain intact.

For example:

```
root@CTM200:/# cmd pcyp msg del 1000 sys_sqi
OK
root@CTM200:/# cmd pcyp msg query 1000
1000:50:$PCYP,…,msgid=1000,sys_uptime=738,*hh<CR><LF>
OK
```

3.4.6 Deleting the complete message

The complete message can also be deleted:

cmd pcyp msg del <msgid>

3.4.7 Replacing a key

PCYP messages are constructed in the order the **pcyp msg add** commands are entered. A user may wish to replace an existing key with a different key keeping all other keys in place, and not having to remove all subsequent keys (ie, key ordering is important).

cmd pcyp msg replace <msgid> <key_old> <key_new>

This will replace a key "key_old" with a key "key_new" in the same location as the existing key

Note, if a key already exists, the key cannot be replaced.

Example:

```
root@CTM200:/# cmd pcyp msg query 1000
1000:60:$PCYP,…,msgid=1000,sys_sqi=48,sys_uptime=738,*hh<CR><LF>
OK
root@CTM200:/# cmd pcyp msg replace 1000 sys_sqi cell_rssi
OK
root@CTM200:/# cmd pcyp msg query 1000
1000:63:$PCYP,…,msgid=1000,cell_rssi=-79,sys_uptime=738,*hh<CR><LF>
OK
```

3.5 Other Commands

3.5.1 Changing the Delimiter

The delimiter command allows the user to change the delimiting character that is inserted between key/value pairs in the PCYP message.

The default delimiter is a comma (,). The user has the freedom to select almost any character desired using the following command:

cmd pcyp delimiter "<char>"
cmd pcyp delimiter '<char>'

Note: Quotes, either double or single are required around the character in order to prevent the shell from interpreting the character.

To view the delimiting character:

```
root@CTM200:/# cmd pcyp delimiter
,
OK
```

Note: Care must be taken when selecting a delimiter. Some characters are invalid, such as backslash (\), asterisk (*), single (') or double quotes (").

3.5.2 Sending a message

When sending the message, PCYP needs a report number defined. For ex, is 1000 is assigned:

```
cmd repaddmes 1 1000
cmd sendreport 1
```

or

```
cmd sendreport 1 1000
```

sends a single report.

To periodically send a report, use a gps condition. For ex, send PCYP 1000 every 30s:

```
cmd gpsaddmes 1 1000
cmd gpscond 1 30
```

Technical Support

Cypress Solutions Service Support Group
1.844.462.9773 or 778.372.4603
9.00am to 5.00pm PST
support@cypress.bc.ca