Numeric Broadcast Protocol (NBP)

Version 1.01 - **2019-05-23**

Author: Weston Pawlowski, HP Tuners LLC
Provided as-is, to promote interoperability between products
Client support in TrackAddict v4.2 and newer (v4.3.6 for latest features)

Features

- Text-based broadcast of numerical data values
- Simple to implement and debug
- Timestamped packets contain one or more data channel updates
- Data is provided as simple channel/unit/value sets, using only numerical values
- Can work over basic serial UARTs, where server/client may not know the connection state
- Can be setup as a one-way broadcast stream; no requirement for server to respond to client requests
- Has provisions to support power-saving features (when using two-way communication)
- Easily recoverable if a client connects mid-stream, or in the event of communication errors
- Supports any number of clients

Broadcast Data Example

```
*NBP1,UPDATE,2356.567
"Battery","V":13.56
"Brake Pedal","%":100.0
#

*NBP1,UPDATE,2356.601
"Steering Wheel","deg":-35.31
"Gear":3
#

@NAME:Test Device
```

The above example shows two packets, each with different relative timestamps (2356.567 and 2356.601 seconds), containing updates for different data channels. They are then followed by an optional metadata line that identifies the device name or model text, “Test Device”.

Broadcast Line Prefix Characters

`*` - Start of Packet
`#` - End of Packet
`@` - Metadata Line (such as device name, version, manufacturer, etc.)
`!` - Client Request (sent from client to server)
`$` - Custom Data Line (manufacturer / device specific data or config; may be sent in either direction)

Broadcast Packet

Basic Rules:

- Each piece must be on its own line, followed by a linefeed character (10 decimal, 0x0A hex, '\n')
- Clients should ignore the carriage return character (13 decimal, 0x0D hex, '\r'), as some servers may inadvertently send it too
- The name and unit of each data channel must not change while the data is being broadcast
- Numerical values (timestamp and data):
    - Must not include any thousands separator (ie use "2356" instead of "2,536")
    - May be either a whole number (integer; 1, 2, 3) or contain decimals (1.2, 2.118, 3.45)
    - Only the '.' character may be used as a decimal separator
    - Some European locales normally use ',' for decimals; that is not supported

Packet Header - Starts a new data packet

Example: `*NBP1,UPDATE,2356.567`

`*` indicates the start of a new packet header

`NBP1` is the packet format ID, version 1

`UPDATE` is the packet type:

- `UPDATE` - Packet is providing updates for one or more data channels
    - Sent whenever new a data value for one or more channels is available
    - This is the proper way to inform clients that new data values are available

- `ALL` - Packet contains a snapshot of all supported data channels and their current values
    - This packet is optional for both server and client, but is recommended to help clients initialize quickly, and to provide them with the full list of channels available. If not implemented, clients would need to observe UPDATE packets over time to discover which data channels are being provided.
    - Data channels must use the same naming and the same units in an ALL packet as they do in UPDATE packets.
    - When two-way communications are in use, it's typical for a client to send a request for this ALL packet when it is initializing.
    - It's recommended that the server send this once every 5 seconds, if that's practical in terms of bandwidth and processing. This will help if either the server or a client is using only one-way communications.

`2356.567` is the relative timestamp, in seconds

- This must always be the same or a higher number than the previous packet sent
- It doesn't need to represent the actual clock / epoch time, but it does need to be an accurate count of seconds relative to itself
- For UPDATE packets, this is the timestamp for the new data; it's helpful for accounting for Bluetooth or other latency

Data Content Line(s) - Provides data as `"channel","unit":value` *-OR-* `"channel":value`

Format 1 Example (with units)
`"Brake Pedal","%":100.0`

`"Brake Pedal"` is the data channel name
- Always enclosed in quotes
- See Appendix A for recommended data channel naming

`"%"` is the unit of measure (a percentage in this case)
- Always enclosed in quotes
- If there's no unit, use "" or choose the format that omits this (see Example 2)
- See Appendix A for recommended data unit naming

`100.0` is the data value; this must always be numerical

Format 2 Example (without units)
`"Gear":3`

`"Gear"` is the data channel name (transmission gear number); always enclosed in quotes

There is no unit of measure for this data channel (see Example 1 to specify a unit)

`3` is the data value (3rd gear); must always be numerical

Packet Footer - Ends the data packet; consists of the '#' character on a new line

Example: `#`

Broadcast Metadata

A metadata line is an optional text string that may be sent periodically to identify the broadcast device. It is both the server and the client's option to implement this or not, and also how much to implement. A server might choose to announce its device type and identity this way at fixed time intervals, or possibly upon detecting a new client (if two-way communications, and the client sends an ALL or KA request).

Example: `@NAME:Test Device`

`@` indicates a broadcast device metadata line

`NAME` is metadata type:
- `NAME` - Display Name (title to be shown to users)
- `MFG` - Manufacturer
- `MODEL` - Model
- `VERSION` - Version
- `SERIAL` - Serial Number

`Test Device` is the text

Client Connection

Clients should attempt to use a two-way connection with the server if possible, however the server has the option of providing only a one-way broadcast connection.

When a client connects to a server, there is no specific initialization or handshaking process. Instead, the client should send a Keep Alive request (if two-way communications) to ensure that the server has not muted its broadcasts to save power.

Upon connecting, the client may very likely start receiving data in the middle of a broadcast packet. Therefore, it should simply ignore all data until it receives a new packet header. It should also do the same if it encounters a communication drop-out or has a parsing error.

To discover which channels are being provided by the server, the client should send an ALL request (if two-way communications), and then listen to the broadcasts for up to 5 seconds, or until it receives an ALL packet, whichever comes first. If the server does not implement the ALL packet (it's recommended but not required), then the client can instead use the unique channels it had observed from the various UPDATE packets received over that time period.

A practical way to implement that is to use a key/value dictionary class or structure to keep track of all unique channel names and their unit labels, from both UPDATE and ALL packet types, until either 5 seconds has elapsed, or an ALL packet has been received and processed.

Client Requests (Two-Way Communications)

This section covers requests that a client may send to the server. It is the server's option if it will listen for these or not.

Requests will be one line each, starting with the `'!'` character, and immediately followed by the request name, and a linefeed character (10 decimal, 0x0A hex, '\n').

Available Requests:

- `ALL` - Requests the server to broadcast the ALL data packet
    - This is useful for discovering all data channels/values upon a new connection

- `KA` - Keep Alive
    - Informs server that a client is connected and it should continue broadcasting
    - Recommended to send this upon client connection, and then once every 15 seconds
    - Useful for servers that have a power saving mode, such as reducing or halting broadcasts when no clients are connected.

- `R1 / R0` - Recording State (v1.01 feature; implemented in TrackAddict v4.3.6)
    - This is a recording synchronization feature. It indicates that the client has started (R1) or stopped (R0) its own recording, and/or it requests that the server or other clients start or stop their recordings.
    - If there are multiple clients, only one of them may send these requests, otherwise it will cause confusion. For that reason, one should be designated as the primary or master.
    - Recommended to send this immediately upon recording state change, and also periodically like "KA" so that it wont be missed in case there was any communication disruption.
    - What the server, or other clients, do with this information is up to them, but common examples are:
        - Synchronizing video cameras, other data recordings, etc.
        - Server device may choose to send data at a more frequent or regular rate
        - Server device may activate certain functionality, modes, displays, etc.

Request Example:

```
!KA
!ALL
```

The above example is typical of when a client first connects to a server. It sends KA to make sure the server knows that it should be broadcasting, followed by ALL to attempt to discover all of the server-provided data channels quickly.

Custom Data Lines

Manufacturer / device specific custom data or configuration lines may also be sent along with this protocol, potentially in either direction, provided that they start with a '$' character and end with a linefeed character (10 decimal, 0x0A hex, '\n').

However, these must never be required for the device to function or for its NBP broadcast data to be understood. Clients and servers that do not understand or support these lines should simply ignore them, and developers implementing this protocol must accommodate those cases.

In order for a client to determine which custom data line format may be in use, it can identify the server's device name/manufacturer/model/version by observing the Broadcast Metadata lines, assuming that the server has opted to broadcast these.

Appendix A: Recommended Data Channel and Unit Naming

With data channels and units being identified by text labels, it becomes useful and sometimes required to use consistent naming, so that different products can easily work together. To that end, the following lists contain some recommended channel and unit names:

Channel Name	Unit(s)	Description
Barometric Pressure	kPa, PSI	Barometer pressure (BARO)
Battery	V	Battery voltage
Brake Pedal	%	Brake pedal position, 0-100 percent
Brake Pressure	kPa, PSI	Brake line pressure
Engine Speed	RPM	Engine Rotations Per Minute (RPM)
Engine Coolant Temp	C, F	Engine Coolant Temperature (ECT)
Engine Oil Temp	C, F	Engine Oil Temperature
Fuel Level	%	Fuel tank level, 0-100 percent
Gear		Transmission gear number (0 for neutral/park, -1 for reverse)
Intake Air Temp	C, F	Intake Air Temperature (IAT)
Intake Mani Pressure	kPa, PSI	Intake Manifold Absolute Pressure (MAP)
Mass Air Flow	g/s, lbs/min	Intake Mass Air Flow (MAF)
Steering Wheel	Deg	Steering wheel position / angle
Throttle Pedal	%	Throttle/Accelerator pedal position, 0-100 percent
Throttle Position	%	Throttle Position Sensor (TPS; typically throttle plate), 0-100
Vehicle Speed	Km/h, MPH	Vehicle Speed (VSS)

Category	Unit(s)
Air/Fuel Ratio	Lambda
Distance / Mileage	Km, miles
Electric Current	A
Electric Potential	V
Electric Power	W, kW
Flow Rate	g/s, lbs/min
Flow Rate	L/h, gal/h
Force / Thrust	N, lb-f
Mass / Weight	kg, lbs
Pressure	kPa, PSI
Pressure (High)	Pa, PSI/1000
Rotational Speed	RPM
Speed	Km/h, MPH
Temperature	C, F
Torque	Nm, ft-lbs