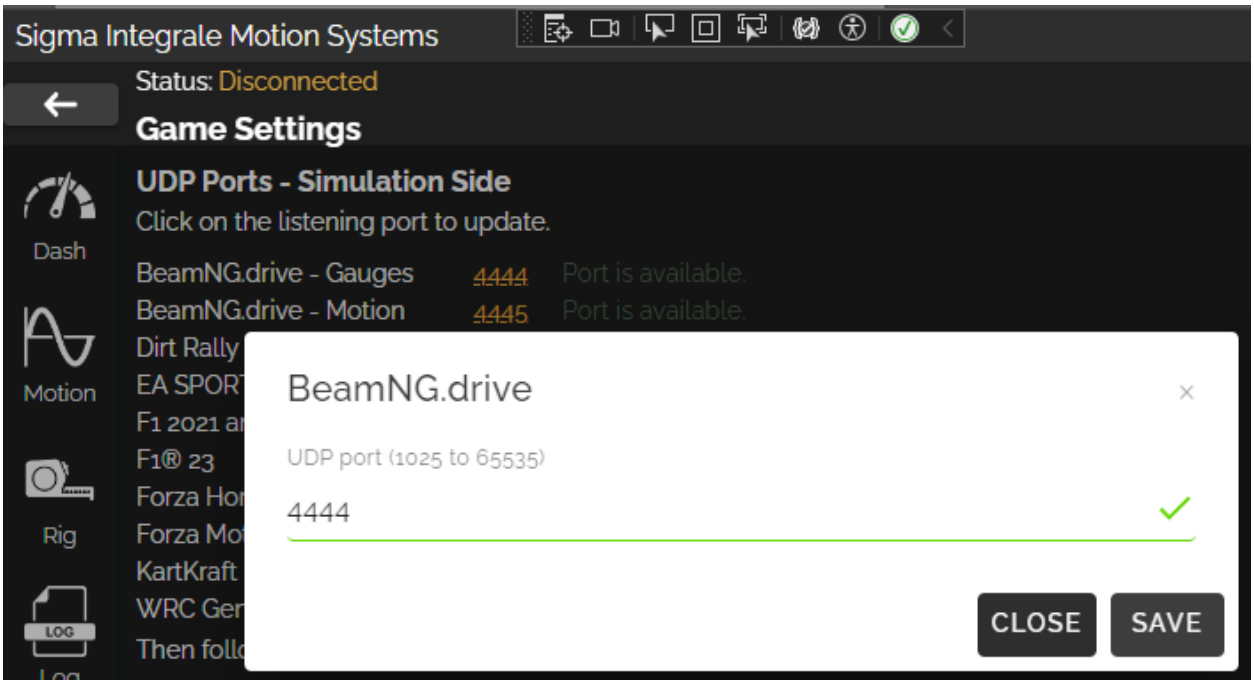
[BeamNG Telemetry Scheme]

BeamNG uses **two** UDP streams to output telemetry:

- Gauges – engine rpm, vehicle speed, etc.
- Motion – surge, sway, heave, pitch, roll, etc.

The DK software expects 300 Hz for the BeamNG telemetry sampling rate, but you can send lower if needed. Sending something other than 300 Hz only affects the smoothing calculation and you can easily change the smoothing values to compensate for a different rate. The higher frequency will benefit the haptics/road-vibrations (i.e. surface texture, suspension impacts) so we recommend sending 300+ Hz. Please do not exceed 400 Hz, which is the highest rate we have tested for stability. A nice rate to use is 333.33 Hz because it's roughly every 3 milliseconds. Try your best to send a steady rate, but we know the rate can easily fluctuate in a Windows OS. When the rate changes momentarily, this is not a big deal as the DK system can easily compensate.

The UDP ports for the two streams above can be modified in the DK software if required. In most cases, there is no need to change the default values.



Gauges Stream

DK only needs two data types from the gauges UDP stream:

- **Vehicle Speed**
    - 0-based byte offset: 12
    - Data type: 4-byte floating point
    - Endianness: little (least significant byte at offset 12, second least at offset 13, third least at offset 14, most significant at offset 15)
    - Unit: meters per second
    - **Note**: If you do not have access to the vehicle speed, it's **critical** to send a fixed value of 0.89408 m/s or higher (i.e. send 1.0 m/s) because DK has a special logic which limits motion when the vehicle reaches a speed below this limit. Motion will not work properly with a fixed value that is less than this threshold.
- **Engine rpm**
    - 0-based byte offset: 16
    - Data type: 4-byte floating point
    - Endianness: little (least significant byte at offset 16, second least at offset 17, third least at offset 18, most significant at offset 19)
    - Note: If you do not have access to engine rpm, you can set this to 0.0.

With this stream, the highest 0-based offset needed is at offset 19 (part of engine rpm), which means the UDP datagram size only needs to be 20 bytes. BeamNG sends a much larger datagram size but it's not necessary to replicate that as DK won't look at data beyond the 20 bytes.

Motion Stream

DK only needs 5 data types from the motion UDP stream:

- **Sway Acceleration**
  - 0-based byte offset: 28
  - Data type: 4-byte float (IEEE 754)
  - Endianness: little (least significant byte at offset 28, second least at offset 29, third least at offset 30, most significant at offset 31)
  - Unit: meters per second-squared (SI unit for acceleration)
  - Direction: positive sway occurs when turning right
- **Surge Acceleration**
  - 0-based byte offset: 32
  - Data type: 4-byte float (IEEE 754)
  - Endianness: little (least significant byte at offset 32, second least at offset 33, third least at offset 34, most significant at offset 35)
  - Unit: meters per second-squared (SI unit for acceleration)
  - Direction: positive surge occurs during forward acceleration
- **Heave Acceleration**
  - 0-based byte offset: 36
  - Data type: 4-byte float (IEEE 754)
  - Endianness: little (least significant byte at offset 36, second least at offset 37, third least at offset 38, most significant at offset 39)
  - Unit: meters per second-squared (SI unit for acceleration)
  - Direction: positive heave occurs when hitting a bump and the vehicle rises

- **Roll Position**
    - 0-based byte offset: 52
    - Data type: 4-byte float (IEEE 754)
    - Endianness: little (least significant byte at offset 52, second least at offset 53, third least at offset 54, most significant at offset 55)
    - Unit: radians
    - Direction: positive roll occurs when turning left
- **Pitch Position**
    - 0-based byte offset: 56
    - Data type: 4-byte float (IEEE 754)
    - Endianness: little (least significant byte at offset 56, second least at offset 57, third least at offset 58, most significant at offset 59)
    - Unit: radians
    - Direction: positive pitch occurs when climbing a hill or during forward acceleration

With this stream, the highest 0-based offset needed is at offset 59 (part of pitch position), which means the UDP datagram size only needs to be 60 bytes. BeamNG sends a much larger datagram size but it's not necessary to replicate that as DK won't look at data beyond the 60 bytes.

Check the C# sample little-endian encoding in the next page. Reach out to us if you need help with achieving the same result in a different language.

Sample Little-Endian Encoding

Here's a sample C# program that packs some of the motion data required in little-endian.

```
{
    // Get the telemetry data here
    float surgeAcceleration = 5.62f; // Offset 28
    float swayAcceleration = 1.34f; // Offset 32
    float heaveAcceleration = 9.8f; // Offset 36

    // Create a UDP packet for packing
    var packet = new byte[60]; // 60 bytes is for the BeamNG motion stream

    // Pack surge acceleration
    var byteData = BitConverter.GetBytes(surgeAcceleration); // Converts float to 4-byte little-endian array
    var index = 28; // Offset
    packet[index++] = byteData[0];
    packet[index++] = byteData[1];
    packet[index++] = byteData[2];
    packet[index++] = byteData[3];

    // Pack sway acceleration
    byteData = BitConverter.GetBytes(swayAcceleration); // Converts float to 4-byte little-endian array
    index = 32; // Offset for sway acceleration
    packet[index++] = byteData[0];
    packet[index++] = byteData[1];
    packet[index++] = byteData[2];
    packet[index++] = byteData[3];

    // Pack heave acceleration
    byteData = BitConverter.GetBytes(heaveAcceleration); // Converts float to 4-byte little-endian array
    index = 36; // Offset for heave acceleration
    packet[index++] = byteData[0];
    packet[index++] = byteData[1];
    packet[index++] = byteData[2];
    packet[index++] = byteData[3];

    // Pack also pitch and roll positions using the same technique

    // Send the UDP packet here
}
```