F1 2020 UDP Specification

By Hoo, May 17 in Technical Assistance

Hoo

Codemasters Staff

Codemasters

163

1,199 posts

Posted May 17

The F1 series of games support the output of certain game data across UDP connections. This data can be used supply race information to external applications, or to drive certain hardware (e.g. motion platforms, force feedback steering wheels and LED devices).

The following information summarises these data structures so that developers of supporting hardware or software are able to configure these to work correctly with the F1 game.

If you cannot find the information that you require, or spot any issues with this specification then please let us know below.

Packet Information

Packet Types

Each packet can now carry different types of data rather than having one packet which contains everything. A header has been added to each packet as well so that versioning can be tracked and it will be easier for applications to check they are interpreting the incoming data in the correct way. Please note that all values are encoded using Little Endian format. All data is packed.

The following data types are used in the structures:

Type	Description
uint8	Unsigned 8-bit integer
int8	Signed 8-bit integer
uint16	Unsigned 16-bit integer
int16	Signed 16-bit integer
float	Floating point (32-bit)
uint64	Unsigned 64-bit integer

Packet Header

```
    uint16    m_packetFormat;             // 2020
    uint8     m_gameMajorVersion;         // Game major version - "X.00"
    uint8     m_gameMinorVersion;         // Game minor version - "1.XX"
    uint8     m_packetVersion;            // Version of this packet type, all start from 1
    uint8     m_packetId;                 // Identifier for the packet type, see below
    uint64    m_sessionUID;               // Unique identifier for the session
    float     m_sessionTime;              // Session timestamp
    uint32    m_frameIdentifier;          // Identifier for the frame the data was retrieved on
    uint8     m_playerCarIndex;           // Index of player's car in the array

   // ADDED IN BETA 2:
    uint8     m_secondaryPlayerCarIndex;  // Index of secondary player's car in the array (splitscreen)
                                          // 255 if no second player
};
```

Packet IDs

The packets IDs are as follows:

Packet Name	Value	Description
Motion	0	Contains all motion data for player's car – only sent while player is in control
Session	1	Data about the session – track, time left
Lap Data	2	Data about all the lap times of cars in the session
Event	3	Various notable events that happen during a session
Participants	4	List of participants in the session, mostly relevant for multiplayer
Car Setups	5	Packet detailing car setups for cars in the race
Car Telemetry	6	Telemetry data for all cars
Car Status	7	Status data for all cars such as damage
Final Classification	8	Final classification confirmation at the end of a race
Lobby Info	9	Information about players in a multiplayer lobby

Motion Packet

The motion packet gives physics data for all the cars being driven. There is additional data for the car being driven with the goal of being able to drive a motion platform setup.

N.B. For the normalised vectors below, to convert to float values divide by 32767.0f – 16-bit signed values are used to pack the data and on the assumption that direction values are always between -1.0f and 1.0f.

Frequency: Rate as specified in menus
Size: 1464 bytes (Packet size updated in Beta 3)
Version: 1

```
struct CarMotionData
{
    float         m_worldPositionX;           // World space X position
    float         m_worldPositionY;           // World space Y position
```

```
    int16         m_worldForwardDirX;         // World space forward X direction (normalised)
    int16         m_worldForwardDirY;         // World space forward Y direction (normalised)
    int16         m_worldForwardDirZ;         // World space forward Z direction (normalised)
    int16         m_worldRightDirX;           // World space right X direction (normalised)
    int16         m_worldRightDirY;           // World space right Y direction (normalised)
    int16         m_worldRightDirZ;           // World space right Z direction (normalised)
    float         m_gForceLateral;            // Lateral G-Force component
    float         m_gForceLongitudinal;       // Longitudinal G-Force component
    float         m_gForceVertical;           // Vertical G-Force component
    float         m_yaw;                      // Yaw angle in radians
    float         m_pitch;                    // Pitch angle in radians
    float         m_roll;                     // Roll angle in radians
};

struct PacketMotionData
{
    PacketHeader    m_header;                   // Header

    CarMotionData   m_carMotionData[22];        // Data for all cars on track

    // Extra player car ONLY data
    float         m_suspensionPosition[4];      // Note: All wheel arrays have the following order:
    float         m_suspensionVelocity[4];      // RL, RR, FL, FR
    float         m_suspensionAcceleration[4];  // RL, RR, FL, FR
    float         m_wheelSpeed[4];              // Speed of each wheel
    float         m_wheelSlip[4];               // Slip ratio for each wheel
    float         m_localVelocityX;             // Velocity in local space
    float         m_localVelocityY;             // Velocity in local space
    float         m_localVelocityZ;             // Velocity in local space
    float         m_angularVelocityX;               // Angular velocity x-component
    float         m_angularVelocityY;           // Angular velocity y-component
    float         m_angularVelocityZ;           // Angular velocity z-component
    float         m_angularAccelerationX;       // Angular velocity x-component
    float         m_angularAccelerationY;           // Angular velocity y-component
    float         m_angularAccelerationZ;       // Angular velocity z-component
    float         m_frontWheelsAngle;           // Current front wheels angle in radians
};
```

Session Packet

The session packet includes details about the current session in progress.

Frequency: 2 per second
Size: 251 bytes (Packet size updated in Beta 3)
Version: 1

```
struct MarshalZone
{
    float  m_zoneStart;   // Fraction (0..1) of way through the lap the marshal zone starts
    int8   m_zoneFlag;    // -1 = invalid/unknown, 0 = none, 1 = green, 2 = blue, 3 = yellow, 4 = red
};

struct WeatherForecastSample
{
    uint8     m_sessionType;                    // 0 = unknown, 1 = P1, 2 = P2, 3 = P3, 4 = Short P, 5
                                                // 6 = Q2, 7 = Q3, 8 = Short Q, 9 = OSQ, 10 = R, 11 = R
                                                // 12 = Time Trial
    uint8     m_timeOffset;                     // Time in minutes the forecast is for
    uint8     m_weather;                        // Weather - 0 = clear, 1 = light cloud, 2 = overcast
                                                // 3 = light rain, 4 = heavy rain, 5 = storm
    int8      m_trackTemperature;               // Track temp. in degrees celsius
    int8      m_airTemperature;                 // Air temp. in degrees celsius
};
```

```
    uint8           m_weather;                  // Weather - 0 = clear, 1 = light cloud, 2 = overcast
                                                // 3 = light rain, 4 = heavy rain, 5 = storm
    int8            m_trackTemperature;         // Track temp. in degrees celsius
    int8            m_airTemperature;           // Air temp. in degrees celsius
    uint8           m_totalLaps;                // Total number of laps in this race
    uint16          m_trackLength;              // Track length in metres
    uint8           m_sessionType;              // 0 = unknown, 1 = P1, 2 = P2, 3 = P3, 4 = Short P
                                                // 5 = Q1, 6 = Q2, 7 = Q3, 8 = Short Q, 9 = OSQ
                                                // 10 = R, 11 = R2, 12 = Time Trial
    int8            m_trackId;                  // -1 for unknown, 0-21 for tracks, see appendix
    uint8           m_formula;                  // Formula, 0 = F1 Modern, 1 = F1 Classic, 2 = F2,
                                                // 3 = F1 Generic
    uint16          m_sessionTimeLeft;          // Time left in session in seconds
    uint16          m_sessionDuration;          // Session duration in seconds
    uint8           m_pitSpeedLimit;            // Pit speed limit in kilometres per hour
    uint8           m_gamePaused;               // Whether the game is paused
    uint8           m_isSpectating;             // Whether the player is spectating
    uint8           m_spectatorCarIndex;        // Index of the car being spectated
    uint8           m_sliProNativeSupport;      // SLI Pro support, 0 = inactive, 1 = active
    uint8           m_numMarshalZones;          // Number of marshal zones to follow
    MarshalZone     m_marshalZones[21];         // List of marshal zones – max 21
    uint8           m_safetyCarStatus;          // 0 = no safety car, 1 = full safety car
                                                // 2 = virtual safety car
    uint8           m_networkGame;              // 0 = offline, 1 = online
    uint8           m_numWeatherForecastSamples; // Number of weather samples to follow
    WeatherForecastSample m_weatherForecastSamples[20];   // Array of weather forecast samples
};
```

Lap Data Packet

The lap data packet gives details of all the cars in the session.

Frequency: Rate as specified in menus
Size: 1190 bytes (Struct updated in Beta 3)
Version: 1

```
struct LapData
{
    float    m_lastLapTime;               // Last lap time in seconds
    float    m_currentLapTime;            // Current time around the lap in seconds

    //UPDATED in Beta 3:
    uint16   m_sector1TimeInMS;           // Sector 1 time in milliseconds
    uint16   m_sector2TimeInMS;           // Sector 2 time in milliseconds
    float    m_bestLapTime;               // Best lap time of the session in seconds
    uint8    m_bestLapNum;                // Lap number best time achieved on
    uint16   m_bestLapSector1TimeInMS;    // Sector 1 time of best lap in the session in milliseconds
    uint16   m_bestLapSector2TimeInMS;    // Sector 2 time of best lap in the session in milliseconds
    uint16   m_bestLapSector3TimeInMS;    // Sector 3 time of best lap in the session in milliseconds
    uint16   m_bestOverallSector1TimeInMS;// Best overall sector 1 time of the session in milliseconds
    uint8    m_bestOverallSector1LapNum;  // Lap number best overall sector 1 time achieved on
    uint16   m_bestOverallSector2TimeInMS;// Best overall sector 2 time of the session in milliseconds
    uint8    m_bestOverallSector2LapNum;  // Lap number best overall sector 2 time achieved on
    uint16   m_bestOverallSector3TimeInMS;// Best overall sector 3 time of the session in milliseconds
    uint8    m_bestOverallSector3LapNum;  // Lap number best overall sector 3 time achieved on


    float    m_lapDistance;               // Distance vehicle is around current lap in metres – could
                                          // be negative if line hasn't been crossed yet
    float    m_totalDistance;             // Total distance travelled in session in metres – could
                                          // be negative if line hasn't been crossed yet
    float    m_safetyCarDelta;            // Delta in seconds for safety car
    uint8    m_carPosition;               // Car race position
```

```
    uint8    m_penalties;               // Accumulated time penalties in seconds to be added
    uint8    m_gridPosition;            // Grid position the vehicle started the race in
    uint8    m_driverStatus;            // Status of driver - 0 = in garage, 1 = flying lap
                                        // 2 = in lap, 3 = out lap, 4 = on track
    uint8    m_resultStatus;            // Result status - 0 = invalid, 1 = inactive, 2 = active
                                        // 3 = finished, 4 = disqualified, 5 = not classified
                                        // 6 = retired
};


struct PacketLapData
{
    PacketHeader    m_header;           // Header

    LapData         m_lapData[22];      // Lap data for all cars on track
};
```

Event Packet

This packet gives details of events that happen during the course of a session.

Frequency: When the event occurs
Size: 35 bytes (Packet size updated in Beta 3)
Version: 1

```
// The event details packet is different for each type of event.
// Make sure only the correct type is interpreted.
union EventDataDetails
{
    struct
    {
        uint8   vehicleIdx; // Vehicle index of car achieving fastest lap
        float   lapTime;    // Lap time is in seconds
    } FastestLap;

    struct
    {
        uint8   vehicleIdx; // Vehicle index of car retiring
    } Retirement;

    struct
    {
        uint8   vehicleIdx; // Vehicle index of team mate
    } TeamMateInPits;

    struct
    {
        uint8   vehicleIdx; // Vehicle index of the race winner
    } RaceWinner;

    struct
    {
        uint8 penaltyType;          // Penalty type – see Appendices
        uint8 infringementType;     // Infringement type – see Appendices
        uint8 vehicleIdx;           // Vehicle index of the car the penalty is applied to
        uint8 otherVehicleIdx;      // Vehicle index of the other car involved
        uint8 time;                 // Time gained, or time spent doing action in seconds
        uint8 lapNum;               // Lap the penalty occurred on
        uint8 placesGained;         // Number of places gained by this
    } Penalty;

    struct
    {
```

```
struct PacketEventData
{
    PacketHeader        m_header;             // Header

uint8                   m_eventStringCode[4]; // Event string code, see below
EventDataDetails        m_eventDetails;       // Event details - should be interpreted differently
                                              // for each type
};
```

Event String Codes

Event	Code	Description
Session Started	“SSTA”	Sent when the session starts
Session Ended	“SEND”	Sent when the session ends
Fastest Lap	“FTLP”	When a driver achieves the fastest lap
Retirement	“RTMT”	When a driver retires
DRS enabled	“DRSE”	Race control have enabled DRS
DRS disabled	“DRSD”	Race control have disabled DRS
Team mate in pits	“TMPT”	Your team mate has entered the pits
Chequered flag	“CHQF”	The chequered flag has been waved
Race Winner	“RCWN”	The race winner is announced
Penalty Issued	“PENA”	A penalty has been issued – details in event
Speed Trap Triggered	“SPTP”	Speed trap has been triggered by fastest speed

Participants Packet

This is a list of participants in the race. If the vehicle is controlled by AI, then the name will be the driver name. If this is a multiplayer game, the names will be the Steam Id on PC, or the LAN name if appropriate.

N.B. on Xbox One, the names will always be the driver name, on PS4 the name will be the LAN name if playing a LAN game, otherwise it will be the driver name.

The array should be indexed by vehicle index.

Frequency: Every 5 seconds
Size: 1213 bytes (Packet size updated in Beta 3)
Version: 1

```
struct ParticipantData
{
    uint8      m_aiControlled;           // Whether the vehicle is AI (1) or Human (0) controlled
    uint8      m_driverId;               // Driver id - see appendix
    uint8      m_teamId;                 // Team id - see appendix
    uint8      m raceNumber;             // Race number of the car
```

```
};

struct PacketParticipantsData
{
    PacketHeader    m_header;           // Header

    uint8           m_numActiveCars;    // Number of active cars in the data - should match number of
                                        // cars on HUD
    ParticipantData m_participants[22];
};
```

Car Setups Packet

This packet details the car setups for each vehicle in the session. Note that in multiplayer games, other player cars will appear as blank, you will only be able to see your car setup and AI cars.

Frequency: 2 per second
Size: 1102 bytes (Packet size updated in Beta 3)
Version: 1

```
struct CarSetupData
{
    uint8     m_frontWing;               // Front wing aero
    uint8     m_rearWing;                // Rear wing aero
    uint8     m_onThrottle;              // Differential adjustment on throttle (percentage)
    uint8     m_offThrottle;             // Differential adjustment off throttle (percentage)
    float     m_frontCamber;             // Front camber angle (suspension geometry)
    float     m_rearCamber;              // Rear camber angle (suspension geometry)
    float     m_frontToe;                // Front toe angle (suspension geometry)
    float     m_rearToe;                 // Rear toe angle (suspension geometry)
    uint8     m_frontSuspension;         // Front suspension
    uint8     m_rearSuspension;          // Rear suspension
    uint8     m_frontAntiRollBar;        // Front anti-roll bar
    uint8     m_rearAntiRollBar;         // Front anti-roll bar
    uint8     m_frontSuspensionHeight;   // Front ride height
    uint8     m_rearSuspensionHeight;    // Rear ride height
    uint8     m_brakePressure;           // Brake pressure (percentage)
    uint8     m_brakeBias;               // Brake bias (percentage)
    float     m_rearLeftTyrePressure;    // Rear left tyre pressure (PSI)
    float     m_rearRightTyrePressure;   // Rear right tyre pressure (PSI)
    float     m_frontLeftTyrePressure;   // Front left tyre pressure (PSI)
    float     m_frontRightTyrePressure;  // Front right tyre pressure (PSI)
    uint8     m_ballast;                 // Ballast
    float     m_fuelLoad;                // Fuel load
};

struct PacketCarSetupData
{
    PacketHeader    m_header;           // Header

    CarSetupData    m_carSetups[22];
};
```

Car Telemetry Packet

This packet details telemetry for all the cars in the race. It details various values that would be recorded on the car such as speed, throttle application, DRS etc.

Frequency: Rate as specified in menus
Size: 1307 bytes (Packet size updated in Beta 3)

```
    uint16    m_speed;                        // Speed of car in kilometres per hour
    float     m_throttle;                     // Amount of throttle applied (0.0 to 1.0)
    float     m_steer;                        // Steering (-1.0 (full lock left) to 1.0 (full lock righ
    float     m_brake;                        // Amount of brake applied (0.0 to 1.0)
    uint8     m_clutch;                       // Amount of clutch applied (0 to 100)
    int8      m_gear;                         // Gear selected (1-8, N=0, R=-1)
    uint16    m_engineRPM;                    // Engine RPM
    uint8     m_drs;                          // 0 = off, 1 = on
    uint8     m_revLightsPercent;             // Rev lights indicator (percentage)
    uint16    m_brakesTemperature[4];         // Brakes temperature (celsius)
    uint8     m_tyresSurfaceTemperature[4];   // Tyres surface temperature (celsius)
    uint8     m_tyresInnerTemperature[4];     // Tyres inner temperature (celsius)
    uint16    m_engineTemperature;            // Engine temperature (celsius)
    float     m_tyresPressure[4];             // Tyres pressure (PSI)
    uint8     m_surfaceType[4];               // Driving surface, see appendices
};

struct PacketCarTelemetryData
{
    PacketHeader        m_header;             // Header

    CarTelemetryData    m_carTelemetryData[22];

    uint32              m_buttonStatus;       // Bit flags specifying which buttons are being pressed
                                              // currently - see appendices

    // Added in Beta 3:
    uint8               m_mfdPanelIndex;      // Index of MFD panel open - 255 = MFD closed
                                              // Single player, race – 0 = Car setup, 1 = Pits
                                              // 2 = Damage, 3 =  Engine, 4 = Temperatures
                                              // May vary depending on game mode
    uint8               m_mfdPanelIndexSecondaryPlayer;   // See above
    int8                m_suggestedGear;      // Suggested gear for the player (1-8)
                                              // 0 if no gear suggested
};
```

Car Status Packet

This packet details car statuses for all the cars in the race. It includes values such as the damage readings on the car.

Frequency: Rate as specified in menus
Size: 1344 bytes (Packet updated in Beta 3)
Version: 1

```
struct CarStatusData
{
    uint8       m_tractionControl;          // 0 (off) - 2 (high)
    uint8       m_antiLockBrakes;           // 0 (off) - 1 (on)
    uint8       m_fuelMix;                  // Fuel mix - 0 = lean, 1 = standard, 2 = rich, 3 = max
    uint8       m_frontBrakeBias;           // Front brake bias (percentage)
    uint8       m_pitLimiterStatus;         // Pit limiter status - 0 = off, 1 = on
    float       m_fuelInTank;               // Current fuel mass
    float       m_fuelCapacity;             // Fuel capacity
    float       m_fuelRemainingLaps;        // Fuel remaining in terms of laps (value on MFD)
    uint16      m_maxRPM;                   // Cars max RPM, point of rev limiter
    uint16      m_idleRPM;                  // Cars idle RPM
    uint8       m_maxGears;                 // Maximum number of gears
    uint8       m_drsAllowed;               // 0 = not allowed, 1 = allowed, -1 = unknown


    // Added in Beta3:
    uint16      m_drsActivationDistance;    // 0 = DRS not available, non-zero - DRS will be available
                                            // in [X] metres
```

```
                                        // F1 Classic - 9 = dry, 10 = wet
                                        // F2 - 11 = super soft, 12 = soft, 13 = medium, 14 = hard
                                        // 15 = wet
    uint8       m_visualTyreCompound;   // F1 visual (can be different from actual compound)
                                        // 16 = soft, 17 = medium, 18 = hard, 7 = inter, 8 = wet
                                        // F1 Classic - same as above
                                        // F2 - same as above
    uint8       m_tyresAgeLaps;         // Age in laps of the current set of tyres
    uint8       m_tyresDamage[4];       // Tyre damage (percentage)
    uint8       m_frontLeftWingDamage;  // Front left wing damage (percentage)
    uint8       m_frontRightWingDamage; // Front right wing damage (percentage)
    uint8       m_rearWingDamage;       // Rear wing damage (percentage)

    // Added Beta 3:
    uint8       m_drsFault;             // Indicator for DRS fault, 0 = OK, 1 = fault

    uint8       m_engineDamage;         // Engine damage (percentage)
    uint8       m_gearBoxDamage;        // Gear box damage (percentage)
    int8        m_vehicleFiaFlags;      // -1 = invalid/unknown, 0 = none, 1 = green
                                        // 2 = blue, 3 = yellow, 4 = red
    float       m_ersStoreEnergy;       // ERS energy store in Joules
    uint8       m_ersDeployMode;        // ERS deployment mode, 0 = none, 1 = medium
                                        // 2 = overtake, 3 = hotlap
    float       m_ersHarvestedThisLapMGUK; // ERS energy harvested this lap by MGU-K
    float       m_ersHarvestedThisLapMGUH; // ERS energy harvested this lap by MGU-H
    float       m_ersDeployedThisLap;   // ERS energy deployed this lap
};

struct PacketCarStatusData
{
    PacketHeader        m_header;       // Header

    CarStatusData       m_carStatusData[22];
};
```

Final Classification Packet

This packet details the final classification at the end of the race, and the data will match with the post race results screen. This is especially useful for multiplayer games where it is not always possible to send lap times on the final frame because of network delay.

Frequency: Once at the end of a race
Size: 839 bytes (Packet size updated in Beta 3)
Version: 1

```
struct FinalClassificationData
{
    uint8     m_position;             // Finishing position
    uint8     m_numLaps;              // Number of laps completed
    uint8     m_gridPosition;         // Grid position of the car
    uint8     m_points;               // Number of points scored
    uint8     m_numPitStops;          // Number of pit stops made
    uint8     m_resultStatus;         // Result status - 0 = invalid, 1 = inactive, 2 = active
                                      // 3 = finished, 4 = disqualified, 5 = not classified
                                      // 6 = retired
    float     m_bestLapTime;          // Best lap time of the session in seconds
    double    m_totalRaceTime;        // Total race time in seconds without penalties
    uint8     m_penaltiesTime;        // Total penalties accumulated in seconds
    uint8     m_numPenalties;         // Number of penalties applied to this driver
    uint8     m_numTyreStints;        // Number of tyres stints up to maximum
    uint8     m_tyreStintsActual[8];  // Actual tyres used by this driver
    uint8     m_tyreStintsVisual[8];  // Visual tyres used by this driver
};
```

```
    uint8                    m_numCars;                // Number of cars in the final classification
    FinalClassificationData  m_classificationData[22];
};
```

Lobby Info Packet

This packet details the players currently in a multiplayer lobby. It details each player's selected car, any AI involved in the game and also the ready status of each of the participants.

Frequency: Two every second when in the lobby
Size: 1169 bytes (Packet size updated in Beta 3)
Version: 1

```
struct LobbyInfoData
{
    uint8     m_aiControlled;           // Whether the vehicle is AI (1) or Human (0) controlled
    uint8     m_teamId;                 // Team id - see appendix (255 if no team currently selected)
    uint8     m_nationality;            // Nationality of the driver
    char      m_name[48];               // Name of participant in UTF-8 format – null terminated
                                        // Will be truncated with ... (U+2026) if too long
    uint8     m_readyStatus;            // 0 = not ready, 1 = ready, 2 = spectating
};

struct PacketLobbyInfoData
{
    PacketHeader    m_header;                     // Header

    // Packet specific data
    uint8               m_numPlayers;             // Number of players in the lobby data
    LobbyInfoData       m_lobbyPlayers[22];
};
```

Restricted data (Your Telemetry setting)

There is some data in the UDP that you may not want other players seeing if you are in a multiplayer game. This is controlled by the "Your Telemetry" setting in the Telemetry options. The options are:

- Restricted (Default) – other players viewing the UDP data will not see values for your car
- Public – all other players can see all the data for your car

Note: You can always see the data for the car you are driving regardless of the setting.

The following data items are set to zero if the player driving the car in question has their "Your Telemetry" set to "Restricted":

Car status packet

- m_fuelInTank
- m_fuelCapacity
- m_fuelMix
- m_fuelRemainingLaps
- m_frontBrakeBias
- m_frontLeftWingDamage
- m_frontRightWingDamage
- m_rearWingDamage
- m_engineDamage
- m_gearBoxDamage
- m_tyresWear (All four wheels)
- m_tyresDamage (All four wheels)
- m_ersDeployMode
- m_ersStoreEnergy
- m_ersDeployedThisLap
- m_ersHarvestedThisLapMGUK
- m_ersHarvestedThisLapMGUH
- m_tyresAgeLaps

Hoo

Codemasters Staff

Codemasters

163

1,199 posts

Posted May 17 (edited)

FAQS

How do I enable the UDP Telemetry Output?

In F1 2020, UDP telemetry output is controlled via the in-game menus. To enable this, enter the options menu from the main menu (triangle / Y), then enter the settings menu - the UDP option will be at the bottom of the list. From there you will be able to enable / disable the UDP output, configure the IP address and port for the receiving application, toggle broadcast mode and set the send rate. Broadcast mode transmits the data across the network subnet to allow multiple devices on the same subnet to be able to receive this information. When using broadcast mode it is not necessary to set a target IP address, just a target port for applications to listen on.

Advanced PC Users: You can additionally edit the game's configuration XML file to configure UDP output. The file is located here (after an initial boot of the game):

```
...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml
```

You should see the tag:

```
<motion>  ...
  <udp enabled="false" broadcast="”false” ip="127.0.0.1" port="20777" sendRate="”20” format="”2020” yourTel
  ...
</motion>
```

Here you can set the values manually. Note that any changes made within the game when it is running will overwrite any changes made manually. Note the enabled flag is now a state.

What has changed since last year?

- F1 2020 sees the following changes to the UDP specification:
- Penalties have been added as a new Event Type
- Weather forecast data is now available in session packets for upcoming sessions
- Reduced the size of the surface and inner tyre temperature fields in the Car Telemetry to reduce overall packet size as more vehicles need to be added
- My Team allows an extra team to race – this means that all the places in the packets where 20 cars were used, 22 are now needed. N.B. this will not be fixed in old formats (2019, 2018, legacy) – if you are in the “My Team” career mode with any format other than 2020 specified, no data will be output. All other game modes will function as before
- Added Vietnamese and Barbadian nationalities
- Added Final Classification packet for end of race results
- Made m_gridPosition the actual numerical position, not 0-based (only in 2020 data)
- Added Lobby Info packet to send data to UDP when in a multiplayer lobby
- Added number of laps the current set of tyres have been used for in status packet
- Split the tyre pressures in Car Setups packet into RL, RR, FL, FR to reflect the game changes
- Added secondary player car index to packet headers for splitscreen
- Added MFD Panel Index to the car telemetry packet for both players
- Added best sector times and lap numbers to the laps packet as they cannot always be recorded correctly (e.g. fast-forwarding time) – N.B. the order has been slightly rearranged to tidy up
- Changed all sector times in lap data packet to be in milliseconds (uint16 instead of float) to reduce packet size so other things can be added
- Added indicator for DRS faults to the car status packet
- Updated ERS mode values
- Speed trap triggered event added
- Added DRS activation distance to indicate whether the car has passed detection and DRS will be available in the subsequent activation zone
- Suggested gear added to the car telemetry packet

```
0 - Rear Left (RL)
1 - Rear Right (RR)
2 - Front Left (FL)
3 - Front Right (FR)
```

Do the vehicle indices change?

During a session, each car is assigned a vehicle index. This will not change throughout the session and all the arrays that are sent use this vehicle index to dereference the correct piece of data.

What encoding format is used?

All values are encoded using Little Endian format.

Are the data structures packed?

Yes, all data is packed, there is no padding used.

Will there always be 20 cars in the data structures?

No, for F1 2020, there is a new feature called “My Team” which allows an extra team to be present on the grid. This means that all previous places where 20 cars were used, 22 is now the maximum. If “My Team” is not active however, most games modes will act as before and have a maximum of 20. Note that if your UDP format is 2019, 2018 or legacy and you are in “My Team” career mode, no UDP output will be produced because of this imitation.

There is still the data item called m_numActiveCars in the participants packet which tells you how many cars are active in the race. However, you should check the individual result status of each car in the lap data to see if that car is actively providing data. If it is not “Invalid” or “Inactive” then the corresponding vehicle index has valid data.

How often are updated packets sent?

For the packets which get updated at “Rate as specified in the menus” you can be guaranteed that on the frame that these get sent they will all get sent together and will never be separated across frames. This of course relies on the reliability of your network as to whether they are received correctly as everything is sent via UDP. Other packets that get sent at specific rates can arrive on any frame.

If you are connected to the game when it starts transmitting the first frame will contain the following information to help initialise data structures on the receiving application:

Packets sent on Frame 1: (All packets sent on this frame have “Session timestamp” 0.000)

- Session
- Participants
- Car Setups
- Lap Data
- Motion Data
- Car Telemetry
- Car Status

As an example, assuming that you are running at 60Hz with 60Hz update rate selected in the menus then you would expect to see the following packets and timestamps:

Packets sent on Frame 2: (All packets sent on this frame have “Session timestamp” 0.016)

- Lap Data
- Motion Data
- Car Telemetry
- Car Status

- Car Setups (since 2 updates per second)
- Lap Data
- Motion Data
- Car Telemetry
- Car Status

Will my old app still work with F1 2020?

F1 2020 uses a new format for the UDP data. However, earlier formats of the data are still supported so that most older apps implemented using the previous data formats should work with little or no change from the developer. To use the old formats, please enter the UDP options menu and set “UDP Format” to either “F1 2019”, “F1 2018” or “Legacy” (for F1 2017 and earlier).

Specifications for the legacy format can be seen here: http://forums.codemasters.com/discussion/53139/f1-2017-d-box-and-udp-output-specification/p1.

Specifications for the F1 2018 format can be seen here: https://forums.codemasters.com/topic/30601-f1-2018-udp-specification/.

How do I enable D-BOX output?

D-BOX output is currently supported on the PC platform. In F1 2020, the D-BOX activation can be controlled via the menus. Navigate to `Game Options->Settings->UDP Telemetry Settings->D-BOX` to activate this on your system.

Advanced PC Users: It is possible to control D-BOX by editing the games’ configuration XML file. The file is located here (after an initial boot of the game):

```
...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml
```

You should see the tag:

```
<motion>
  <dbox enabled="false" />
  …
</motion>
```

Set the “`enabled`” value to “`true`” to allow the game to output to your D-BOX motion platform. Note that any changes made within the game when it is running will overwrite any changes made manually.

How can I disable in-game support for LED device?

The F1 game has native support for some of the basic features supported by some external LED devices, such as the *Leo Bodnar SLI Pro* and the *Fanatec* steering wheels. To avoid conflicts between Codemasters’ implementation and any third-party device managers on the PC platform it may be necessary to disable the native support. This is done using the following `led_display` flags in the `hardware_settings_config.xml`. The file is located here (after an initial boot of the game):

```
...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml
```

The flags to enabled/disable LED output are:

```
<led_display fanatecNativeSupport="true" sliProNativeSupport="true" />
```

Please note there is an additional flag to manually control the LED brightness on the SLI Pro:

```
<led_display sliProForceBrightness="127" />
```

This option (using value in the range 0-255) will be ignored when setting the sliProNativeSupport flag to "false".

Also note it is now possible to edit these values on the fly via the Game Options->Settings->UDP Telemetry Settings menu.

Can I configure the UDP output using an XML File?

PC users can edit the game's configuration XML file to configure UDP output. The file is located here (after an initial boot of the game):

```
...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml
```

You should see the tag:

```
<motion>
  ...
  <udp enabled="false" broadcast="”false”" ip="127.0.0.1" port="20777" sendRate="”20”" format="”2020”" yourTel
  ...
</motion>
```

Here you can set the values manually. Note that any changes made within the game when it is running will overwrite any changes made manually.

Edited June 24 by Hoo
Updated list of changes for this year

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted May 17

Appendices

Here are the values used for the team ID, driver ID and track ID parameters.

N.B. Driver IDs in network games differ from the actual driver IDs. All the IDs of human players start at 100 and are unique within the game session, but don't directly correlate to the player.

Team IDs

ID	Team	ID	Team	ID	Team
0	Mercedes	21	Red Bull 2010	42	Art GP '19
1	Ferrari	22	Ferrari 1976	43	Campos '19
2	Red Bull Racing	23	ART Grand Prix	44	Carlin '19
3	Williams	24	Campos Vexatec Racing	45	Sauber Junior Charouz '19
4	Racing Point	25	Carlin	46	Dams '19

7	Haas	28	Russian Time	49	Prema '19
8	McLaren	29	MP Motorsport	50	Trident '19
9	Alfa Romeo	30	Pertamina	51	Arden '19
10	McLaren 1988	31	McLaren 1990	53	Benetton 1994
11	McLaren 1991	32	Trident	54	Benetton 1995
12	Williams 1992	33	BWT Arden	55	Ferrari 2000
13	Ferrari 1995	34	McLaren 1976	56	Jordan 1991
14	Williams 1996	35	Lotus 1972		
15	McLaren 1998	36	Ferrari 1979		
16	Ferrari 2002	37	McLaren 1982		
17	Ferrari 2004	38	Williams 2003		
18	Renault 2006	39	Brawn 2009		
19	Ferrari 2007	40	Lotus 1978		
20	McLaren 2008	41	F1 Generic car	255	My Team

Driver IDs

ID	Driver	ID	Driver	ID	Driver
0	Carlos Sainz	37	Peter Belousov	70	Rashid Nair
1	Daniil Kvyat	38	Klimek Michalski	71	Jack Tremblay
2	Daniel Ricciardo	39	Santiago Moreno	74	Antonio Giovinazzi
6	Kimi Räikkönen	40	Benjamin Coppens	75	Robert Kubica
7	Lewis Hamilton	41	Noah Visser	78	Nobuharu Matsushita
9	Max Verstappen	42	Gert Waldmuller	79	Nikita Mazepin
10	Nico Hulkenburg	43	Julian Quesada	80	Guanya Zhou
11	Kevin Magnussen	44	Daniel Jones	81	Mick Schumacher
12	Romain Grosjean	45	Artem Markelov	82	Callum Ilott
13	Sebastian Vettel	46	Tadasuke Makino	83	Juan Manuel Correa
14	Sergio Perez	47	Sean Gelael	84	Jordan King

19	Lance Stroll	50	George Russell	87	Anthoine Hubert
20	Arron Barnes	51	Maximilian Günther	88	Guiliano Alesi
21	Martin Giles	52	Nirei Fukuzumi	89	Ralph Boschung
22	Alex Murray	53	Luca Ghiotto		
23	Lucas Roth	54	Lando Norris		
24	Igor Correia	55	Sérgio Sette Câmara		
25	Sophie Levasseur	56	Louis Delétraz		
26	Jonas Schiffer	57	Antonio Fuoco		
27	Alain Forest	58	Charles Leclerc		
28	Jay Letourneau	59	Pierre Gasly		
29	Esto Saari	62	Alexander Albon		
30	Yasar Atiyeh	63	Nicholas Latifi		
31	Callisto Calabresi	64	Dorian Boccolacci		
32	Naota Izum	65	Niko Kari		
33	Howard Clarke	66	Roberto Merhi		
34	Wilheim Kaufmann	67	Arjun Maini		
35	Marie Laursen	68	Alessio Lorandi		
36	Flavio Nieves	69	Ruben Meijer		

Track IDs

ID	Track
0	Melbourne
1	Paul Ricard
2	Shanghai
3	Sakhir (Bahrain)
4	Catalunya
5	Monaco
6	Montreal

9	Hungaroring
10	Spa
11	Monza
12	Singapore
13	Suzuka
14	Abu Dhabi
15	Texas
16	Brazil
17	Austria
18	Sochi
19	Mexico
20	Baku (Azerbaijan)
21	Sakhir Short
22	Silverstone Short
23	Texas Short
24	Suzuka Short
25	Hanoi
26	Zandvoort

Nationality IDs

ID	Nationality	ID	Nationality	ID	Nationality
1	American	31	Greek	61	Panamanian
2	Argentinean	32	Guatemalan	62	Paraguayan
3	Australian	33	Honduran	63	Peruvian
4	Austrian	34	Hong Konger	64	Polish
5	Azerbaijani	35	Hungarian	65	Portuguese
6	Bahraini	36	Icelander	66	Qatari
7	Belgian	37	Indian	67	Romanian
8	Bolivian	38	Indonesian	68	Russian

11	Bulgarian	41	Italian	71	Scottish
12	Cameroonian	42	Jamaican	72	Serbian
13	Canadian	43	Japanese	73	Singaporean
14	Chilean	44	Jordanian	74	Slovakian
15	Chinese	45	Kuwaiti	75	Slovenian
16	Colombian	46	Latvian	76	South Korean
17	Costa Rican	47	Lebanese	77	South African
18	Croatian	48	Lithuanian	78	Spanish
19	Cypriot	49	Luxembourger	79	Swedish
20	Czech	50	Malaysian	80	Swiss
21	Danish	51	Maltese	81	Thai
22	Dutch	52	Mexican	82	Turkish
23	Ecuadorian	53	Monegasque	83	Uruguayan
24	English	54	New Zealander	84	Ukrainian
25	Emirian	55	Nicaraguan	85	Venezuelan
26	Estonian	56	North Korean	86	Welsh
27	Finnish	57	Northern Irish	87	Barbadian
28	French	58	Norwegian	88	Vietnamese
29	German	59	Omani		
30	Ghanaian	60	Pakistani		

Surface types

These types are from physics data and show what type of contact each wheel is experiencing.

ID	Surface
0	Tarmac
1	Rumble strip
2	Concrete
3	Rock
4	Gravel

7	Grass
8	Water
9	Cobblestone
10	Metal
11	Ridged

Button flags

These flags are used in the telemetry packet to determine if any buttons are being held on the controlling device. If the value below logical ANDed with the button status is set then the corresponding button is being held.

Bit Flag	Button
0x0001	Cross or A
0x0002	Triangle or Y
0x0004	Circle or B
0x0008	Square or X
0x0010	D-pad Left
0x0020	D-pad Right
0x0040	D-pad Up
0x0080	D-pad Down
0x0100	Options or Menu
0x0200	L1 or LB
0x0400	R1 or RB
0x0800	L2 or LT
0x1000	R2 or RT
0x2000	Left Stick Click
0x4000	Right Stick Click

Penalty types

1	Stop Go
2	Grid penalty
3	Penalty reminder
4	Time penalty
5	Warning
6	Disqualified
7	Removed from formation lap
8	Parked too long timer
9	Tyre regulations
10	This lap invalidated
11	This and next lap invalidated
12	This lap invalidated without reason
13	This and next lap invalidated without reason
14	This and previous lap invalidated
15	This and previous lap invalidated without reason
16	Retired
17	Black flag timer

Infringement types

ID	Infringement meaning
0	Blocking by slow driving
1	Blocking by wrong way driving
2	Reversing off the start line
3	Big Collision
4	Small Collision
5	Collision failed to hand back position single
6	Collision failed to hand back position multiple
7	Corner cutting gained time
8	Corner cutting overtake single

11	Ignoring blue flags
12	Ignoring yellow flags
13	Ignoring drive through
14	Too many drive throughs
15	Drive through reminder serve within n laps
16	Drive through reminder serve this lap
17	Pit lane speeding
18	Parked for too long
19	Ignoring tyre regulations
20	Too many penalties
21	Multiple warnings
22	Approaching disqualification
23	Tyre regulations select single
24	Tyre regulations select multiple
25	Lap invalidated corner cutting
26	Lap invalidated running wide
27	Corner cutting ran wide gained time minor
28	Corner cutting ran wide gained time significant
29	Corner cutting ran wide gained time extreme
30	Lap invalidated wall riding
31	Lap invalidated flashback used
32	Lap invalidated reset to track
33	Blocking the pitlane
34	Jump start
35	Safety car to car collision
36	Safety car illegal overtake
37	Safety car exceeding allowed pace
38	Virtual safety car exceeding allowed pace
39	Formation lap below allowed speed

42	Safety car falling too far back
43	Black flag timer
44	Unserved stop go penalty
45	Unserved drive through penalty
46	Engine component change
47	Gearbox change
48	League grid penalty
49	Retry penalty
50	Illegal time gain
51	Mandatory pitstop

2

Drospy

D

Members
5
34 posts

Posted May 20

Hi,

is there any possibility to have "DELTA TIME" in UDP telemetry ?

Thank you

Walter

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted May 20

> On 5/20/2020 at 10:01 AM, Drospy said:
>
> Hi,
>
> is there any possibility to have "DELTA TIME" in UDP telemetry ?
>
> Thank you
>
> Walter

This isn't something we plan on doing as delta time implementation is a design choice that involves specific logic to evaluate rather than an objective piece of real-time data. There are too many different approaches to this to give a "correct" delta, so we will simply provide the time and positional data for the cars and app developers can work out how they want to interpret this.

UP100

Posted May 20

Moderator

Moderator

1,085
2,868 posts

> Is there any possibility to have "DELTA TIME" in UDP telemetry ?
>
> Thank you
>
> Walter

Is there something you'd need delta time for? Maybe there's something else that could provide you the functionality that you require.

Pawel567

Members
20
61 posts

Posted May 20 (edited)

> On 5/20/2020 at 1:47 PM, UP100 said:
>
> I would introduce for exceeding the delta instead of the penalty of passing through boxes a penalty of 5-10 seconds for a stop & go penalty that will take place in the box or will be added to you

I would have introduced for crossing the delta instead of a 5-10 sec penalty stop & go penalty which will take place in boxing or it will be added to you

Edited May 20 by Pawel567

Drospy

D

Members
5
34 posts

Posted May 20

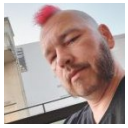
> On 5/20/2020 at 1:47 PM, UP100 said:
>
> Is there something you'd need delta time for? Maybe there's something else that could provide you the functionality that you require.

Now, calculate delta time between sector time and best time for sector. I think that this is a good info.

However, this doesn't work in Time Trial, because we don't have best time for sector

---

EnsiFerrum



Members
84
730 posts

Posted May 20

I need the index of the current shown HUD page.

- CarSetupPanel -> Index = 0
- PitSetupPanel -> Index = 1
- CarDamagePanel -> Index = 2
- CarEnginePanel -> Index = 3
- CarTemperaturePanel -> Index = 4

Any possibilty to get this info?

---

cjorgens79

C

Members
3

Posted May 21

Just a minor point, I have been doing some api testing and have noticed that the two extra slots (21, 22) have randomish values in them for some fields that could make it a little confusing for some people working with the api. For example, the api is reporting 20 cars in the race (as expected), however slot 21 has semi valid x,z positional data, race position is 1 and lap number is 1. Slot 22 has no positional data, its race position is 0, but it is on lap 1. It would be nice if these slots could be cleared (zeroed out) properly so there is no randomish data in them to make it clearer for integrators when dealing with the telemetry data.

Oasis81

Members
4
49 posts

Posted May 22 (edited)

HI

same 2019 the data stops before the finish time in Q.

is it possible to extend the data sent until all the cars have finish the session?

second and most important thing for me:

the sectors: in the 2019 the sectors are erased after the lap, because at the new sector 1 in the new lap overwrite the sector 1 of the last lap.

this happen in every case, if the next lap is the pit entry or if it's a worst lap than the previous.

is it possible that the new sector overwrite the old sector only if the new lap is better than the previous?

Edited May 22 by Oasis81

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted May 22

> On 5/22/2020 at 6:15 PM, Oasis81 said:
>
> HI
>
> same 2019 the data stops before the finish time in Q.
>
> is it possible to extend the data sent until all the cars have finish the session?
>
> second and most important thing for me:
>
> the sectors: in the 2019 the sectors are erased after the lap, because at the new sector 1 in the new lap overwrite the sector 1 of the last lap.
>
> this happen in every case, if the next lap is the pit entry or if it's a worst lap than the previous.
>
> is it possible that the new sector overwrite the old sector only if the new lap is better than the previous?

Hi @Oasis81 ,

In single player mode the session ends once the player has finished their session, so the UDP data simply reflects what the game is doing. If the game were to allow the session to continue and the player to spectate for the rest of the session (like it does in multiplayer) then this should just work in the UDP data. However, this is a bigger feature request that is beyond the scope of the UDP telemetry data. I can pass the suggestion back to the team if that is what you are proposing.

For the second point, the LapData packet currently keeps only the latest sector information. These times can obviously be stored by your app if needed. Are you suggesting that we should add a set of "best sector" times into the LapData packet?

Thanks,
Hoo.

Oasis81

Members
4
49 posts

Posted May 22 (edited)

> On 5/22/2020 at 6:50 PM, Hoo said:
>
> Hi @Oasis81 ,
>
> In single player mode the session ends once the player has finished their session, so the UDP data simply reflects what the game is doing. If the game were to allow the session to continue and the player to spectate for the rest of the session (like it does in multiplayer) then this should just work in the UDP data. However, this is a bigger

> into the LapData packet?
>
> Thanks,
> Hoo.

Hello Hoo

thanks for the answer.

For the first point yes, the data stops when the player ends the Q. but usually there are some cars in the track that have no finisched their laps, and this could be modify the classify. My suggestion is the choice to the player to stops immediatly the Q same as now, or to choose to continue to follow the Q in spectator mode same as multy, until all the cars are over the Q time, to have the UDP data also for this time left.

it will be also more realistic, because maybe in the Q I have the pole position, but when the Q finished at the next screen I'll see that i'v lost it, but I couldn't followed this in real time.

for my view.

For the second point.

My app atm works same as this: every car has a personal list of record that stores every sector and every lap, order the list with the better lap and select the first record. , after this match this data with other list with other cars time and export the classify. this data can be stored only in real time mode,

ex:

car 1 | lap 3 | time 3 | sect 1.3 | sect 2.3 | sect 3.3

car 1 | lap 2 | time 2 | sect 1.2 | sect 2.2 | sect 3.2

car 1 | lap 1 | time 1 | sect 1.1 | sect 2.1 | sect 3.1

---

car 2 | lap 2 | time 2 | sect 1.2 | sect 2.2 | sect 3.2

car 2 | lap 1 | time 1 | sect 1.1 | sect 2.1 | sect 3.1

Class

car 1 | time 3 | sect 1.3 | sect 2.3 | sect 3.3

car 2 |time 2 | sect 1.2 | sect 2.2 | sect 3.2

ecc...

but this has 1 big problem:

1) if the player skip something, same as the pit exit or the return to the box or i FFW the time and other cars are on track and does their laps, , the data sent from the game about the sectors could be skipped, so i can't store it.

so, my suggestion is STORE the Table that the player seen on the monitor when is at the box , where is all the cars with their better time and the respective sectors.

because the game can store this data in some way, also if i skip something or FFW the time, but i can't.

I can suggest something like this:

m_bestlap (there is)

m_bestlapSector1

m_bestlapSector2

m_bestlapSector3

or only sect 1 and 2 could be fine.

thanks

regards

Edited May 22 by Oasis81

1

cjorgens79

C

Members
3
168 posts

Posted May 23

@Hoo

The LapData packet has some anomalous data in it. This is the current state of drivers 6 seconds into a 30 minute practice session.

Note that the Lap Distance and Total Distance values are large negatives. I would have expected these to be 0, as the session has just started and each driver is in the garage. The Pit Status also seems a bit random too, again everyone is in the garage however the pit status seems random between them. The very first LapData packet of the session looks exactly the same as well.

D1 -> NIEVES	sLapDist = -4152.5244140625	sTotalDist = -4152.5244140625	sRacePos = 6	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D2 -> FOREST	sLapDist = -4238.380859375	sTotalDist = -4238.380859375	sRacePos = 2	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D3 -> MICHALSKI	sLapDist = -4138.3212890625	sTotalDist = -4138.3212890625	sRacePos = 18	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D4 -> CALABRESI	sLapDist = -4209.7451171875	sTotalDist = -4209.7451171875	sRacePos = 13	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D5 -> ATIYEH	sLapDist = -4202.4130859375	sTotalDist = -4202.4130859375	sRacePos = 7	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D6 -> CLARKE	sLapDist = -4195.6396484375	sTotalDist = -4195.6396484375	sRacePos = 10	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D7 -> LEVASSEUR	sLapDist = -4252.5053710938	sTotalDist = -4252.5053710938	sRacePos = 12	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D8 -> BELOUSOV	sLapDist = -4159.8955078125	sTotalDist = -4159.8955078125	sRacePos = 11	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D9 -> CORREIA	sLapDist = -4245.1743164063	sTotalDist = -4245.1743164063	sRacePos = 15	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D10 -> IZUMI	sLapDist = -4188.3076171875	sTotalDist = -4188.3076171875	sRacePos = 8	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D11 -> MURRAY	sLapDist = -4259.23046875	sTotalDist = -4259.23046875	sRacePos = 3	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D12 -> SCHIFFER	sLapDist = -4231.0493164063	sTotalDist = -4231.0493164063	sRacePos = 17	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D13 -> KAUFMANN	sLapDist = -4167.0986328125	sTotalDist = -4167.0986328125	sRacePos = 5	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D14 -> MORENO	sLapDist = -4145.6923828125	sTotalDist = -4145.6923828125	sRacePos = 4	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D15 -> ROTH	sLapDist = -4266.5610351563	sTotalDist = -4266.5610351563	sRacePos = 9	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D16 -> SAARI	sLapDist = -4223.8271484375	sTotalDist = -4223.8271484375	sRacePos = 14	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D17 -> GILES	sLapDist = -4302.1782226563	sTotalDist = -4302.1782226563	sRacePos = 19	lapNum = 1	mPitStatus: 1	sDriverStatus: 0
D18 -> LAURSEN	sLapDist = -4174.4619140625	sTotalDist = -4174.4619140625	sRacePos = 20	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D19 -> LETOURNEAU	sLapDist = -4216.4951171875	sTotalDist = -4216.4951171875	sRacePos = 1	lapNum = 1	mPitStatus: 0	sDriverStatus: 0
D20 -> BARNES	sLapDist = -4294.8837890625	sTotalDist = -4294.8837890625	sRacePos = 16	lapNum = 1	mPitStatus: 1	sDriverStatus: 0

cjorgens79

C

Members
3
168 posts

Posted May 23

> On 5/22/2020 at 6:50 PM, Hoo said:
>
> Hi @Oasis81 ,
>
> In single player mode the session ends once the player has finished their session, so the UDP data simply reflects what the game is doing. If the game were to allow the session to continue and the player to spectate for the rest of the session (like it does in multiplayer) then this should just work in the UDP data. However, this is a bigger feature request that is beyond the scope of the UDP telemetry data. I can pass the suggestion back to the team if that is what you are proposing.
>
> For the second point, the LapData packet currently keeps only the latest sector information. These times can obviously be stored by your app if needed. Are you suggesting that we should add a set of "best sector" times into the LapData packet?
>
> Thanks,
> Hoo.

I think this could be useful as well for those single player sessions where time jumps occur by going back to garage or jumping to a flying lap. There is plenty of room in the packet to allow the addition of 2 more floats for

LonelyRacer

Posted May 23

L

Members
8
60 posts

> On 5/23/2020 at 3:15 AM, cjorgens79 said:
>
> @Hoo
>
> The LapData packet has some anomalous data in it. This is the current state of drivers 6 seconds into a 30 minute practice session.
>
> Note that the Lap Distance and Total Distance values are large negatives. I would have expected these to be 0, as the session has just started and each driver is in the garage. The Pit Status also seems a bit random too, again everyone is in the garage however the pit status seems random between them. The very first LapData packet of the session looks exactly the same as well.

D1 -> NIEVES	sLapDist = -4152.5244140625	sTotalDist = -4152.5244140625	sRacePos = 6	lapNum = 1	mPitStatus: 0	sDriverStatus
D2 -> FOREST	sLapDist = -4238.380859375	sTotalDist = -4238.380859375	sRacePos = 2	lapNum = 1	mPitStatus: 1	sDriverStatus
D3 -> MICHALSKI	sLapDist = -4138.3212890625	sTotalDist = -4138.3212890625	sRacePos = 18	lapNum = 1	mPitStatus: 0	sDriverStatus
D4 -> CALABRESI	sLapDist = -4209.7451171875	sTotalDist = -4209.7451171875	sRacePos = 13	lapNum = 1	mPitStatus: 0	sDriverStatus
D5 -> ATIYEH	sLapDist = -4202.4130859375	sTotalDist = -4202.4130859375	sRacePos = 7	lapNum = 1	mPitStatus: 0	sDriverStatus
D6 -> CLARKE	sLapDist = -4195.6396484375	sTotalDist = -4195.6396484375	sRacePos = 10	lapNum = 1	mPitStatus: 0	sDriverStatus
D7 -> LEVASSEUR	sLapDist = -4252.5053710938	sTotalDist = -4252.5053710938	sRacePos = 12	lapNum = 1	mPitStatus: 1	sDriverStatus
D8 -> BELOUSOV	sLapDist = -4159.8955078125	sTotalDist = -4159.8955078125	sRacePos = 11	lapNum = 1	mPitStatus: 0	sDriverStatus
D9 -> CORREIA	sLapDist = -4245.1743164063	sTotalDist = -4245.1743164063	sRacePos = 15	lapNum = 1	mPitStatus: 1	sDriverStatus
D10 -> IZUMI	sLapDist = -4188.3076171875	sTotalDist = -4188.3076171875	sRacePos = 8	lapNum = 1	mPitStatus: 0	sDriverStatus
D11 -> MURRAY	sLapDist = -4259.23046875	sTotalDist = -4259.23046875	sRacePos = 3	lapNum = 1	mPitStatus: 1	sDriverStatus
D12 -> SCHIFFER	sLapDist = -4231.0493164063	sTotalDist = -4231.0493164063	sRacePos = 17	lapNum = 1	mPitStatus: 1	sDriverStatus
D13 -> KAUFMANN	sLapDist = -4167.0986328125	sTotalDist = -4167.0986328125	sRacePos = 5	lapNum = 1	mPitStatus: 0	sDriverStatus
D14 -> MORENO	sLapDist = -4145.6923828125	sTotalDist = -4145.6923828125	sRacePos = 4	lapNum = 1	mPitStatus: 0	sDriverStatus
D15 -> ROTH	sLapDist = -4266.5610351563	sTotalDist = -4266.5610351563	sRacePos = 9	lapNum = 1	mPitStatus: 1	sDriverStatus
D16 -> SAARI	sLapDist = -4223.8271484375	sTotalDist = -4223.8271484375	sRacePos = 14	lapNum = 1	mPitStatus: 0	sDriverStatus
D17 -> GILES	sLapDist = -4302.1782226563	sTotalDist = -4302.1782226563	sRacePos = 19	lapNum = 1	mPitStatus: 1	sDriverStatus
D18 -> LAURSEN	sLapDist = -4174.4619140625	sTotalDist = -4174.4619140625	sRacePos = 20	lapNum = 1	mPitStatus: 0	sDriverStatus
D19 -> LETOURNEAU	sLapDist = -4216.4951171875	sTotalDist = -4216.4951171875	sRacePos = 1	lapNum = 1	mPitStatus: 0	sDriverStatus
D20 -> BARNES	sLapDist = -4294.8837890625	sTotalDist = -4294.8837890625	sRacePos = 16	lapNum = 1	mPitStatus: 1	sDriverStatus

The negative values mean, the car is that far of from the start/finish line. F1 2018 & F1 2019 this was the case with P & Q. So when you start your outlap, the values increase hitting 0 when your outlap finishes and the first real lap starts.

And the Trash in "unused" cars is also, what has been there at least in time trial and online races. If it doesn't get fixed/changed, then you have to massage the data yourself. That is what I have been doing with earlier games. I.e. by using the sResultStatus <= 1 to detect, if the driver is active and then modify inactive's values accordingly so they don't mess with the proper data.

The issue I see with the zeroing to"0" the values is that there is DriverID 0 for Carlos Sainz, so if they "0" all data, you still have the issue if the driver is "inactive" or if it is Sainz. I would prefer they change the Sainz driverId to something else. Would make many things easier.

I still need to do more driving and looking at the data to see, if there is anything odd/new going on this year. Thus far (some 3 hours of driving + some 15 hours or running save data feed on my tool), haven't noticed anything too critical.

BernoAU

Posted May 24

B

Members
2
16 posts

It would be great if the LapData struct included a time gap to the car ahead.

All we need is the gap to car ahead as obviously the gap to car behind is the car ahead gap of the next position down. Also if we want to show the gap to the leader we can simply add up all the gaps ahead. We also already have the current lap number so can calculate if a car is a lap down.

So we could replicate the leaderboard with time gaps with just one extra float per driver in the struct.

Members
3
168 posts

> The negative values mean, the car is that far of from the start/finish line. F1 2018 & F1 2019 this was the case with P & Q. So when you start your outlap, the values increase hitting 0 when your outlap finishes and the first real lap starts.
>
> And the Trash in "unused" cars is also, what has been there at least in time trial and online races. If it doesn't get fixed/changed, then you have to massage the data yourself. That is what I have been doing with earlier games. I.e. by using the sResultStatus <= 1 to detect, if the driver is active and then modify inactive's values accordingly so they don't mess with the proper data.
>
> The issue I see with the zeroing to"0" the values is that there is DriverID 0 for Carlos Sainz, so if they "0" all data, you still have the issue if the driver is "inactive" or if it is Sainz. I would prefer they change the Sainz driverId to something else. Would make many things easier.
>
> I still need to do more driving and looking at the data to see, if there is anything odd/new going on this year. Thus far (some 3 hours of driving + some 15 hours or running save data feed on my tool), haven't noticed anything too critical.

Thanks for the reply, after I posted it occurred to me that was probably the reason why the distances were negative. The pit status is a bug though as you have mentioned, it is easy enough to work around though.

Re the "zero" stuff, yeah i know what you mean about the conflicting ids, however the result status field's intention is to indicate whether or not the driver slot has valid data or not, if it is not a valid driver then any data in that driver's packet should be ignored. I found this flag after that original post, however i still think it is cleaner for the unused slots to be zeroed out.

DaveyGravy

Posted May 24

D

Members
10
53 posts

> On 5/24/2020 at 2:45 AM, BernoAU said:
>
> It would be great if the LapData struct included a time gap to the car ahead.
>
> All we need is the gap to car ahead as obviously the gap to car behind is the car ahead gap of the next position down. Also if we want to show the gap to the leader we can simply add up all the gaps ahead. We also already have the current lap number so can calculate if a car is a lap down.
>
> So we could replicate the leaderboard with time gaps with just one extra float per driver in the struct.

This would be super useful. I'm currently doing my own calculations to arrive at the time gaps which is OK, but direct data from the game would be much better and more accurate.

On a different note, it'd be nice to be able to know if DRS was going to be available soon, ie. the detection line has been crossed and will be available in the DRS zone. As far as I can tell this isn't covered.

Oasis81

Posted May 25 (edited)

O

Members
4
49 posts

a question

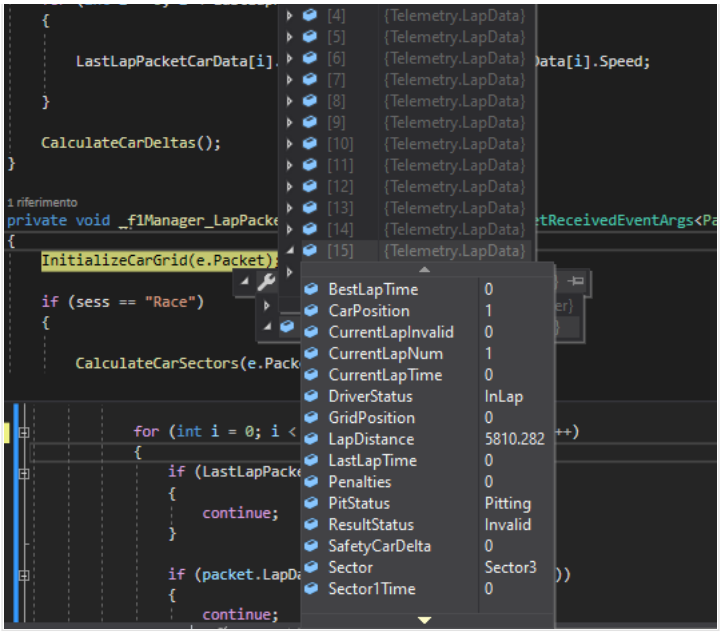
it's only mine problem or there is an error when you go on Q2 about the data?

the car with index 15 has CarPosition at 1, and my app generate errors because there are 2 cars with CarPosition at 1., the first and the 15

the problem is that the car is inactive because in the Q2 the Car 15 is out of qualify, so, sam as other cars under it, the car position should be 0, not 1.

with f1 2019 I haven't this problem.

edit: after some checks i see that the first car out of range (15 in Q2) ... 11 in Q3 ... 21 in Race has always Car position at 1, and not at 0.



Edited May 25 by Oasis81

cjorgens79

Members
3
168 posts

Posted May 26

> On 5/25/2020 at 8:48 PM, Oasis81 said:

a question

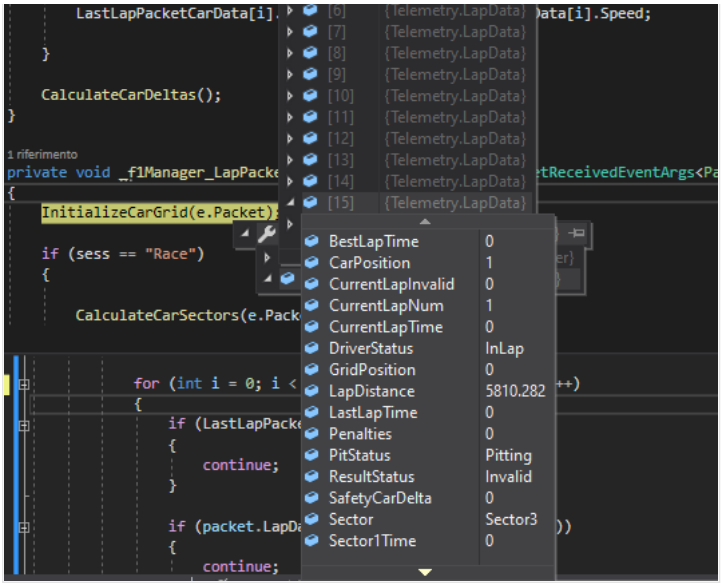
it's only mine problem or there is an error when you go on Q2 about the data?

the car with index 15 has CarPosition at 1, and my app generate errors because there are 2 cars with CarPosition at 1., the first and the 15

the problem is that the car is inactive because in the Q2 the Car 15 is out of qualify, so, sam as other cars under it, the car position should be 0, not 1.

with f1 2019 I haven't this problem.

edit: after some checks i see that the first car out of range (15 in Q2) ... 11 in Q3 ... 21 in Race has always Car position at 1, and not at 0.



Check the m_resultStatus field, its intention is to indicate whether or not the driver slot has valid data or not, if it is not a valid driver then any data in that driver's packet should be ignored. So basically ignore any drivers who's m_resultStatus value is less than 2.

Oasis81

O

Members
4
49 posts

Posted May 26

> On 5/26/2020 at 12:57 AM, cjorgens79 said:
>
> Check the m_resultStatus field, its intention is to indicate whether or not the driver slot has valid data or not, if it is not a valid driver then any data in that driver's packet should be ignored. So basically ignore any drivers who's m_resultStatus value is less than 2.

HI

Yes I must use that field now, the problem i that i must rewrite more and more part of the app because the packet comes with all the data, and without that carposition for invalid cars it was more easy to do, without filtering before.

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted May 26

Thanks for all of the feedback so far everyone. We're looking into some of these issues now.

In terms of delta times, we're not considering adding this in at the moment as the data should exist for everyone to do what they need. As stated above, delta time has so many different approaches and uses that it isn't something that we can add in to accommodate everyone's own use case. For example, there have been requests for the gap to the car in front, pitting delta, realtime delta used in Time Trial. Even something a simple as gap to the car in front comes with lots of design interpretation to make it work correctly:

If we simply work out the gap time gap between each car passing the specified point in the circuit, how do we interpolate between differences in track position reported by the car as these don't usually align? What happens when cars enter the pit lane or go off track? Do we consider using predictive deltas, or based them retrospectively on cars passing the same point in the track? What happens if you are not racing the car in front due to lapping them or alternative pitting strategies? What happens if cars are on in-laps or out-laps? ... and so on.

The game already uses a few different approaches to delta times and even more have been suggested by the community. As each one is a subjective design choice, we prefer to just provide the lap and time data and let you all choose how to use this.

We'll provide an update on the various changes and bug fixes ahead of the next beta phase. Thanks.

2

Members
8
60 posts

Just wanted to share this, as I see so many people asking about the Delta. And I know there are few ways to do this.

I used to calculate the delta as an estimate based on average speed and distances between the cars. Wasn't very good.

I know that some people store only leaders time and position and calculate the delta from that. This is pretty easy way to get the deltas.

My solution

This is what I have done in my tool: https://www.racedepartment.com/downloads/telemetry-application.27456/

The users can set the accuracy, basically the distance between detection points. Range is from 1m to 500m.

At the start of the race, I create an empty matrix, where each cell will contain <driverId, totalTime>. i.e. DeltaItem[columns][cars].

The width of the matrix (i.e. columns of the matrix) is: totalLapsInRace * trackLength / accuracyDistance. For 5 lap race in 5k track with 100m accuracy the matrix width is 250 columns. With 20 cars, it means 5000 cells, each containing an int and float, so approx 50kb of memory. For a 70lap race in 5k track with 1m accuracy, memory used is about 70mb. For 10 hour endurance with 60 cars (in ACC), this would be around .3gb, but there e.g. 200m accuracy is more than sufficient, which takes about 1.5mb to store the whole 10h race.

At the start, the first column contains the situation at the start of the race. (totalDistanceDriven <= 0).

The index for each column after start is easily calculated with floor(totalDistanceDriven / accuracyDistance) + 1. So if accuracy is 50, you have driven 225, then the column is 5.

As the race commences and cars pass to the new "column", I store the leader to the top of the column and cars behind to lower positions, i.e. each column then contains from the top down the track positions at that point during the race. All cells in that column contain the driverId + totalTime at that point.

Note, all times use the totalTime, you can get that .e.g from the header's sessionTime.

So then when I calculate delta to the leader for certain car, I take their totalDistanceDriven, find the column. In that column I take their cell (based on driverId), then deduct from the total time the leaders total time (at row 0). This is the delta to the leader. E.g. leader entered column 6 at 17.1 and you entered there at 17.8, so delta is .7.

Delta to the car in front you get by finding the cars column (with totalDistanceDriven) + row (based on driverId), then calculating the totalTime diff for the row and row - 1. So you entered at column 6, your total time entering the range was 17.8. The car in front (your row - 1) entered the column at 17.6, so the delta to front is .2.

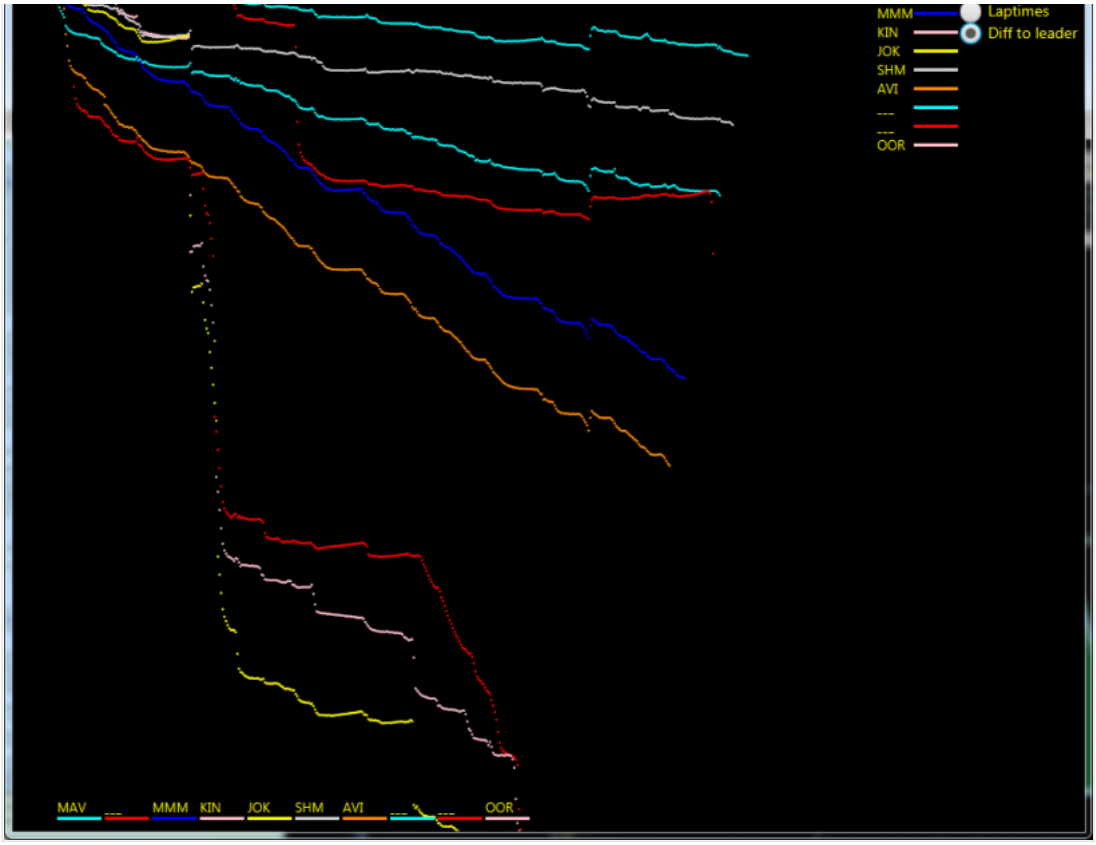
Delta to the car behind is similar, you just find the totalDistanceDriven for car behind, their correct column and then calculate the time difference to your cell on that column. The car behind is at column 5. Your time entering that range was 17.1 and the car behind entered at 17.5, so the delta to back is .4

Getting the data is pretty fast, as calls are direct array calls with index. The only small exception is to find the row, you compare with, which needs extra loop to find the right row with corresponding driverId, but in average it is MAX_CARS/2, so with F1 series about 10 extra calls per search. But if you really need to optimize that too, you could store for the columnsr a hashmap<driverId, index_in_that_column> to reduce this search also to a single call, but at a cost of memory (columns * hashmap size)

One optimization one could do, is to store the data at the "game" order, i..e car at index 0 is stored at row 0 and so on. With F1 series this should be ok, as the order stays same during the race, but there are other games, where the "game order" changes, as people join/drop. This optimization would take away the driverId search mentioned above.

Other optimization for a dash solution would be to store only 4 rows, 1 for leader, 1 for car in front, 1 for the player and 1 for the car behind. Then you would always have leader at row 0, person in front (at that point) at row 1, the player in row 2 and the car behind in row 3. So the "id checking" would go away, but you would still know, who is in front/behind, if you save the driverId on the cells. And you would have access to the delta history to leader for the player. For 70lap on 6k track with 1m accuracy takes about 17mb.

With this implementation, during the race and/or at the end of the race, I can produce a chart like this.



Cheers.

Drospy

D

Members
5
34 posts

Posted May 27

> On 5/26/2020 at 10:23 AM, Hoo said:
>
> Thanks for all of the feedback so far everyone. We're looking into some of these issues now.
>
> In terms of delta times, we're not considering adding this in at the moment as the data should exist for everyone to do what they need. As stated above, delta time has so many different approaches and uses that it isn't something that we can add in to accommodate everyone's own use case. For example, there have been requests for the gap to the car in front, pitting delta, realtime delta used in Time Trial. Even something a simple as gap to the car in front comes with lots of design interpretation to make it work correctly:
>
> If we simply work out the gap time gap between each car passing the specified point in the circuit, how do we interpolate between differences in track position reported by the car as these don't usually align? What happens when cars enter the pit lane or go off track? Do we consider using predictive deltas, or based them retrospectively on cars passing the same point in the track? What happens if you are not racing the car in front due to lapping them or alternative pitting strategies? What happens if cars are on in-laps or out-laps? ... and so on.
>
> The game already uses a few different approaches to delta times and even more have been suggested by the community. As each one is a subjective design choice, we prefer to just provide the lap and time data and let you all choose how to use this.
>
> We'll provide an update on the various changes and bug fixes ahead of the next beta phase. Thanks.

Ok, all right, just a request:
In Time Trial there isn't data of bestlap, only best lap time, I ask to add SectorTime data of best lap.

Thank you for your support

1 2 3 4 5 6 NEXT » Page 1 of 7

GO TO TOPIC LISTING
Technical Assistance

Home › F1 Games › F1® 2020 Game Forum › Technical Assistance ›

All Activity