D-BOX SDK OVERVIEW

A document describing content and concepts from D-BOX Live Motion SDK

CONFIDENTIAL
226-989-0006-EN1

CONTENT

What is in this document?

D-BOX SDK is used to build gateways between simulation software applications as well as video game titles and D-BOX Motion Codes. This document is a description of D-BOX's Live Motion SDK.

It contains a glossary of the ICD (interface control document) used to exchange data and events between a third party software application and D-BOX Motion Codes.

It also contains an appendix of typical code samples to support a fast learning curve for integrators.

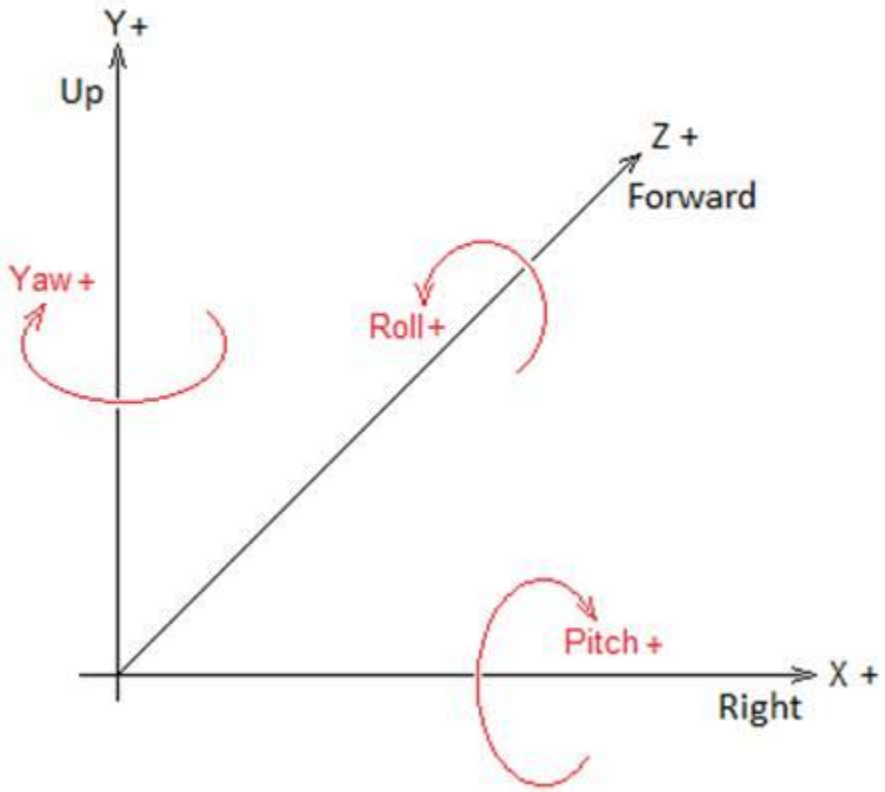
INITIAL REMARKS

Telemetry dataset

The telemetry data we need will depend on the nature of your software: race simulation, construction simulation, flight simulation, etc…

Left handed Cartesian coordinate system

Note that the SDK uses a left-Handed Cartesian coordinate system as shown below:



METHODS

Structure layout registration macros

Use these macros to define the fields layout, type and meaning in a struct. They will create a static GetStructInfo() method in your struct suitable for registration with the RegisterEvent method. Later on, an instance of the registered struct can be supplied to the PostEvent method.

Sample usage: (note that there is no ";" at end of macro calls)

```
struct SampleConfig {
        int EngineMaxRpm;
        float EngineMaxPower;
        DBOX_STRUCTINFO_BEGIN()
                DBOX_STRUCTINFO_FIELD(SampleConfig, EngineMaxRpm, dbox::FT_INT32, dbox::FM_ENGINE_RPM_MAX)
                DBOX_STRUCTINFO_FIELD(SampleConfig, EngineMaxPower, dbox::FT_FLOAT32,
dbox::FM_ENGINE_POWER_MAX)
        DBOX_STRUCTINFO_END()
};
```

Then SampleConfig::GetStructInfo() can be supplied to RegisterEvent method.

WARNING: Your StructInfo layout must match exactly the layout of your struct. These macros facilitate this because they can both be maintained close to each other in your source code.

SDK methods

dbox::LiveMotion::Initialize(PCCHAR sAppKey, U32 nAppBuildVersion)

- Global initialization. It must be called once at start-up. Initialize/Terminate calls must be balanced.
- Parameter:
  - sAppKey: should be set to a constant char * that uniquely identifies your application (usually assigned by D-BOX).
  - nAppBuildVersion: should be set to an integer representing your build number. This field may be used by D-BOX to track version changes or updates in your application.

dbox::LiveMotion::Terminate()

- Global termination. It must be called at the end and it is very important to do so.
- Initialize/Terminate calls must be balanced.

dbox::LiveMotion::Open(U8 nSessionCount = 1)

- Opens the motion output device. This will normally prepare the motion platform for output. Open/Close calls must be balanced
- Parameters:
  - nSessionCount: number of sessions requested to be opened.

    For applications supporting 1 motion output, default value of 1 should be used. For application supporting multiple motion outputs, this parameter must be set to the supported number of motion outputs.

dbox::LiveMotion::Close()

- Closes the motion output device. This will normally release the motion platform and immediately stop any pending output.
- Open/Close calls must be balanced.

dbox::LiveMotion::Start()

- Gently starts motion activity based on PostEvent calls and internal state of Motion Code logic. A fade-in will be applied for smooth transition. It should be called as soon as the end-user can interact with the virtual environment, like start of simulation/level, resume from pause, etc.

dbox::LiveMotion::Stop()

- Gently stops motion activity. A fade-out will be applied for smooth transition. It should be called as soon as the end-user can no longer interact with the virtual environment, like end of simulation/level, pause, etc.

dbox::LiveMotion::ResetState()

- Resets the internal state of the Motion Code logic. It should normally be called before Start or after Stop (except for pauses) when the environment or context changed. If IsStarted() is true, this equivalent sequence will be executed: Stop, ResetState, Start.

dbox::LiveMotion::RegisterEvent(U32 nPermanentEventKey, EEventMeaning eMeaning, const StructInfo & oGetStructInfo)

- Registers an event or message with related data or attributes. Events are used to communicate configuration or real-time data that can be used for motion generation. You can combine many data fields in a structure and post them at once in a single event. To do so you must create a structure and use the

DBOX_STRUCTINFO_* macros to create structure layout information that will be used by this RegisterEvent method.

- This way, the D-BOX Motion Code logic will be able to interpret your structure data on subsequent PostEvent calls. To post these events you will need to use the 2-parameter PostEvent method.
- Parameters:
  - nPermanentEventKey: This is your unique reference for later posting this event. Once a Permanent Event Key is used in motion generation, it must not change with future releases of your application to prevent compatibility breaks.
  - eMeaning: This is the general meaning of your event.
  - oGetStructInfo: For a structure with appropriate DBOX_STRUCTINFO_* layout, use [your struct name]::GetStructInfo().

dbox::LiveMotion::RegisterEvent(U32 nPermanentEventKey, EEventMeaning eMeaning)

- Registers a simple event, trigger without related data, or attribute.
- To post these events you will need to use the 1-parameter PostEvent method.

dbox::LiveMotion::PostEvent() {return PostEvent(nPermanentEventKey, &oData, sizeof(T));}

- Sends an event and a related data structure from your application to all sessions. You must supply your registered Permanent Event Key and a related structure instance reference.

dbox::LiveMotion::PostEvent(U32 nPermanentEventKey)

- Sends an event from your application to all sessions, without related data or attributes. This is useful for small events using only your registered Permanent Event Key.

dbox::LiveMotion::PostEventToSession(U8 nTargetSession, U32 nPermanentEventKey, const T & oData) {return PostEventToSession(nTargetSession, nPermanentEventKey, &oData, sizeof(T));};

- Sends an event and a related data structure from your application to a specific session. You must supply your registered Permanent Event Key and a related structure instance reference.

dbox::LiveMotion::PostEventToSession(U8 nTargetSession, U32 nPermanentEventKey)

- Sends an event from your application to a specific session, without related data or attributes.
- This is useful for small events using only your registered Permanent Event Key.

dbox::LiveMotion::IsInitialized()

- Returns true if the API has successfully initialized.

dbox::LiveMotion::IsOpened()

- Returns true if the API has initialized and has successfully opened.

dbox::LiveMotion::IsStarted()

- Returns true if the API has initialized, opened and started.

DATA FIELDS

The following is the list of known meanings that can be associated to fields. This association will be used during event registration. D-BOX may add meanings to this list over time, but should not remove any.

Fields types

- FRAME FIELDS

  Frame Fields should be posted at every simulation frame or game loop frame.

  Their state is global and will be maintained by the Motion Code between frames. Even if a same FRAME FIELD is registered in different event structures, it will update the same global state in the Motion Code.

- CONFIGURATION FIELDS

  Configuration fields can be posted when required or when a change occurs.

  Their state is global and will be maintained by the Motion Code between updates. Even if a same CONFIGURATION FIELD is registered in different event structures, it will update the same global state in the Motion Code.

Global Mixer Fields

- FM_MASTER_GAIN_DB : (configuration field) Master gain applied at the final output, in decibel. It is recommended to expose this setting in the application's user interface. (0dB is the default level).
- FM_MASTER_SPECTRUM_DB : (configuration field) Motion/Vibration balance control, in decibel. It is recommended to expose this setting in the application's user interface. Value must be between -20dB and +20dB: -20dB being maximum vibration (high frequencies) attenuation, +20dB being maximum motion (low frequencies) attenuation and 0dB being normal signal (full spectrum).

General Dynamics fields

- FM_ELAPSED_TIME_MS : (frame field) Milliseconds elapsed since the last post of this field. Required when physics are computed ahead of time to synchronize all FRAME fields in time.
- FM_ACTOR_GFORCE_XYZ : (frame field) Xyz vector of 3D felt G-Force vector in G relative to actor (i.e. driver/pilot) position and orientation. Positive X is right acceleration, Positive Y is up acceleration, and Positive Z is forward acceleration. Normally standing on Earth: Y = +1.0.
  Note: ACTOR_GFORCE is always relative to actor and not [Reference] (see below).

Reference is a 3D point in space where physics are computed. References can be actors or vehicles.

1. When a Reference is an Actor (i.e. driver, pilot, soldier).
   - Reference positive X is right-hand side of actor body, positive Y is upward and positive Z is forward.
   - No ACTOR_OFFSET required but must be 0 if supplied.

   i.e. If the [Reference] for given acceleration and velocity is the pilot then no ACTOR_OFFSET is needed.

2. When a Reference is a Vehicle (i.e. car, plane, carrier)
   - Reference positive X is right side of vehicle looking forward, positive Y is upward and positive Z is forward.
   - ACTOR_OFFSET required specifying actor position and orientation relative to vehicle reference.

   i.e. If the [Reference] for given acceleration and velocity is the vehicle's center of gravity, ACTOR_OFFSET should represent the offset of the pilot from said center of gravity.

- FM_ACTOR_OFFSET_POSITION_XYZ : (configuration field) Xyz vector of actor (i.e. driver/pilot) position in meters, relative to the [Reference] used. Offset position must be specified in the case where [Reference] position is different from actor position. Positive X is right position, Positive Y is up position and Positive Z is forward position, relative to the [Reference] local XYZ axis.

- FM_ACTOR_OFFSET_YAW_RAD : (configuration field) Right yaw or heading of actor (i.e. driver/pilot), in radians, relative to [Reference] local Y-axis. In the case where the actor orientation is different from [Reference] orientation, that angle should be specified. Yaw expresses a rotation along [Reference] Y-axis (vehicle up). 0 means actor facing front of the vehicle, Pi/2 means actor facing right of the vehicle.
- FM_ACTOR_OFFSET_PITCH_RAD : (configuration field) Forward pitch of actor (i.e. driver/pilot), in radians, relative to [Reference] X-Z plane. In the case where the actor orientation is different from [Reference] orientation, that angle should be specified. Pitch expresses a rotation along actor X-axis, after Yaw is applied. 0 means actor same level as vehicle, pi/2 means actor facing vehicle floor.
- FM_ACTOR_OFFSET_ROLL_RAD : (configuration field) Left roll of actor (i.e. driver/pilot), in radians, relative to [Reference] pitch. In the case where the actor orientation is different from [Reference] orientation, that angle should be specified. Roll expresses a rotation along actor Z-axis, after Pitch and Yaw are applied. 0 means lateral level, pi/2 means actor rotated left.
- FM_ACCELERATION_XYZ : (frame field) Xyz vector of 3D acceleration in m/s^2 relative to a [Reference] position and orientation. If actor position/orientation is different from [Reference] position/orientation, ACTOR_OFFSET fields must be specified. Positive X is right acceleration, Positive Y is up acceleration, Positive Z is forward acceleration.
- FM_VELOCITY_XYZ : (frame field) Xyz vector of 3D velocity in m/s relative to a [Reference] position and orientation. If actor position/orientation is different from [Reference] position/orientation, ACTOR_OFFSET fields must be specified. Positive X is right velocity, Positive Y is up velocity, Positive Z is forward velocity.
- FM_POSITION_XYZ : (frame field) Xyz vector of position of a [Reference] in the world. Positive X is to the east, Positive Y is to the sky and Positive Z is to the north.
- FM_YAW_RAD : (frame field) Right yaw or heading of a [Reference], in radians, relative to world Y axis (0 means toward North, pi/2 means toward East). If actor orientation is different from [Reference] orientation, ACTOR_OFFSET fields must be specified. Yaw expresses a rotation along [Reference] Y-axis.
- FM_PITCH_RAD : (frame field) Forward pitch of a [Reference], in radians, relative to world X-Z plane (0 means forward level, pi/2 means toward ground). If actor orientation is different from [Reference] orientation, ACTOR_OFFSET fields must be specified. Pitch expresses a rotation along [Reference] X-axis, after Yaw is applied.
- FM_ROLL_RAD : (frame field) Left roll of a [Reference], in radians, relative to pitch (0 means lateral level, pi/2 means body rotated left). If actor orientation is different from [Reference] orientation, ACTOR_OFFSET fields must be specified. Roll expresses a rotation along [Reference] Z-axis, after Pitch and Yaw are applied

- FM_ANGULAR_ACCELERATION_XYZ : (frame field) Xyz structure of 3D angular acceleration in rad/s^2 relative to a [Reference] position and orientation. If actor position/orientation is different from [Reference] position/orientation, ACTOR_OFFSET fields must be specified. Positive X is forward pitch acceleration, Positive Y is right yaw acceleration, Positive Z is left roll acceleration.
- FM_ANGULAR_VELOCITY_XYZ : (frame field) Xyz structure of 3D angular velocity in rad/s relative to a [Reference] position and orientation. If actor position/orientation is different from [Reference] position/orientation, ACTOR_OFFSET fields must be specified. Positive X is forward pitch velocity, Positive Y is right yaw velocity, Positive Z is left roll velocity.
- FM_DYNAMIC_ROLL_GAIN_DB : (configuration field) User configurable gain for left-right general dynamics motion effect. This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.
- FM_DYNAMIC_PITCH_GAIN_DB : (Configuration field) User configurable gain for front-back general dynamics motion effect. This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.
- FM_DYNAMIC_HEAVE_GAIN_DB : (configuration field) User configurable gain for up-down general dynamics motion effect. This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.

General Context fields

- FM_VEHICLE : (Configuration field) Numeric identification of current vehicle.
- FM_VEHICLE_STR : (Configuration field) String identification of current vehicle.
- FM_VEHICLE_TYPE : (Configuration field) Numeric identification of current vehicle type or class.
- FM_VEHICLE_TYPE_STR : (Configuration field) String identification of current vehicle type or class.
- FM_MAP : (Configuration field) Custom numeric identification of current map, track or level.
- FM_MAP_STR : (Configuration field) Custom string identification of current map, track or level.
- FM_MAP_TYPE : (configuration field) Custom numeric identification of the type or classification of current map, track or level.
- FM_MAP_TYPE_STR : (configuration field) Custom string identification of the type or classification of type of map, track or level.

General Vehicle Engine fields

- FM_ENGINE_RPM_MAX : (configuration field) Absolute maximum RPM of vehicle engine.
- FM_ENGINE_RPM_RED : (configuration field) Common start of red RPM zone of vehicle engine.
- FM_ENGINE_RPM_IDLE : (configuration field) Common idle RPM of vehicle engine.
- FM_ENGINE_RPM : (frame field) Instantaneous engine rotation speed in RPM.
- FM_ENGINE_CYLINDERS : (configuration field) Number of cylinders in main engine.
- FM_ENGINE_OFFSET_XYZ : (configuration field) Xyz vector of engine offset relative to [Reference].
- FM_ENGINE_POWER_MAX : (configuration field) Maximum engine power in Watts (1 HP = 745.699872 watts).
- FM_ENGINE_POWER : (frame field) Instantaneous engine power in Watts (1 HP = 745.699872 watts).
- FM_ENGINE_TORQUE_MAX : (configuration field) Maximum engine torque in Newton meters (1 foot-pound = 1.35581795 newton-meter).
- FM_ENGINE_TORQUE: (frame field) Instantaneous engine torque in Newton meters (1 foot-pound = 1.35581795 newton-meter). Torque should be negative when braking on engine compression.

- FM_ENGINE_TRANSMISSION_GEAR : (frame field) Current transmission gear number (Reverse = -1, Neutral = 0, 1st = 1, 2nd = 2...).
- FM_ENGINE_TRANSMISSION_RPM : (frame field) Current transmission shaft rotation speed in RPM.
- FM_ENGINE_BOOST : (frame field) Engine current boost intensity from 0.0 to 1.0. 0.0 is no boost (normal), 1.0 is maximum boost effect.
- FM_ENGINE_GAIN_DB : (configuration field) User configurable gain for engine vibration effect. This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.
- FM_ENGINE_BOOST_GAIN_DB : (configuration field) User configurable gain for engine boost vibration effect (nitrous). This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.

Vehicle Multi-Engine fields (i.e. aircraft)

- FM_ENGINE_COUNT : (configuration field) Engine count, up to 8. If not specified, only one engine is assumed.
- FM_ENGINE1_RPM : (frame field) Instantaneous engine 1 rotation speed in RPM.
- FM_ENGINE1_OFFSET_XYZ : (configuration field) Xyz vector of engine 1 offset relative to [Reference].
- FM_ENGINE2_RPM: (frame field) Instantaneous engine 2 rotation speed in RPM.
- FM_ENGINE2_OFFSET_XYZ : (configuration field) Xyz vector of engine 2 offset relative to [Reference].
- FM_ENGINE3_RPM : (frame field) Instantaneous engine 3 rotation speed in RPM.
- FM_ENGINE3_OFFSET_XYZ : (configuration field) Xyz vector of engine 3 offset relative to [Reference].
- FM_ENGINE4_RPM : (frame field) Instantaneous engine 4 rotation speed in RPM.
- FM_ENGINE4_OFFSET_XYZ : (configuration field) Xyz vector of engine 4 offset relative to [Reference].
- FM_ENGINE5_RPM : (frame field) Instantaneous engine 5 rotation speed in RPM.
- FM_ENGINE5_OFFSET_XYZ : (configuration field) Xyz vector of engine 5 offset relative to [Reference].
- FM_ENGINE6_RPM : (frame field) Instantaneous engine 6 rotation speed in RPM.
- FM_ENGINE6_OFFSET_XYZ : (configuration field) Xyz vector of engine 6 offset relative to [Reference].
- FM_ENGINE7_RPM : (frame field) Instantaneous engine 7 rotation speed in RPM.
- FM_ENGINE7_OFFSET_XYZ : (configuration field) Xyz vector of engine 7 offset relative to [Reference].
- FM_ENGINE8_RPM : (frame field) Instantaneous engine 8 rotation speed in RPM.
- FM_ENGINE8_OFFSET_XYZ : (configuration field) Xyz vector of engine 8 offset relative to [Reference].

Vehicle Corners fields (independent wheel info)

- FM_SUSPENSION_TRAVEL_MAX_QUAD : (configuration field) Absolute maximum down suspension travel range in meters (1 inch = 0.0254 meters). If applicable, this should also include frame torsion the vehicle can normally handle.
- FM_SUSPENSION_TRAVEL_NORM_QUAD : (frame field) Current suspension travel position relative to maximum travel range, from 0.0 (in air) to 1.0 (full down).
- FM_SUSPENSION_TRAVEL_QUAD : (frame field) Current suspension travel position, from 0.0 (in air) to down SUSPENSION_TRAVEL_MAX in meters (1 inch = 0.0254 meters).
- FM_SUSPENSION_GAIN_DB : (configuration field) User configurable gain for road texture motion effect (based on suspension or tire load). This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.
- FM_TIRE_LOAD_NORM_QUAD : (frame field) Load for each tire relative to vehicle weight, normalized to 1.0 at rest, 0.0 when in air.
- FM_TIRE_SLIP_RATIO_QUAD : (frame field) Longitudinal slip velocity ratio, generally defined as ((A * R) - V) / V) where A = tire angular velocity in rad/s, R = tire radius in meters and V = ground velocity in meters. Ratio should be 0 when V = 0 and A = 0. Ratio should be limited to a positive value (i.e. +5) when V = 0 and A != 0. Ratio = 0 means 100% grip and |ratio| > 1.0 means loss of grip.
- FM_TIRE_SLIP_ANGLE_RAD_QUAD : (frame field) Angle between a rolling wheel's actual direction of travel and the direction towards which it is pointing. The slip angle is per tire and is relative to the tire itself and not the vehicle.
- FM_TIRE_SLIP_VELOCITY_QUAD : (frame field) Current tangent slip velocity for each tire in m/s (1 mph = 0.44704 meters / second). This should represent the speed difference magnitude between tire and surface. Can be expressed as ((A * R) – V) where A = tire angular velocity in rad/s, R = tire radius in meters and V = ground velocity in meters.
- FM_TIRE_SLIP_GAIN_DB : (configuration field) User configurable gain for tire skid motion effect. This setting can be exposed in the application's user interface. Value should be between -60dB and +20dB, 0dB being normal signal. Values lower than -60 means no effect, use -999 for explicit mute.

Aircraft fields

- FM_AIRCRAFT_TAS_KT : (frame field) True Airspeed, in knots (1 mph = 0.86898 knot).
- FM_AIRCRAFT_VFE_KT : (configuration field) Maximum Flaps Extended Speed Vfe, in knots (1 mph = 0.86898 knot).
- FM_AIRCRAFT_VLO_KT : (configuration field) Maximum Landing Gear operation speed, in knots (1 mph = 0.86898 knot)
- FM_AIRCRAFT_VS_KT : (configuration field) Stalling speed with flaps retracted Vs, in knots (1 mph = 0.86898 knot).
- FM_AIRCRAFT_VS0_KT : (configuration field) Stalling speed in landing configuration Vs0 (flaps fully extended, landing gear deployed), in knots (1 mph = 0.86898 knot).
- FM_AIRCRAFT_CRITICAL_AOA_DEG : (configuration field) Critical Angle of Attack (AOA, alpha), angle in degrees where aircraft is said to be in a stall.
- FM_AIRCRAFT_AOA_DEG : (frame field) Angle of Attack (AOA, alpha), angle in degrees between the chord line of the wing and the direction of the relative wind.
- FM_AIRCRAFT_AGL_FT : (frame field) Altitude above ground level in feet.
- FM_AIRCRAFT_ON_GROUND : (frame field) Specifies whether aircraft is on ground.
- FM_AIRCRAFT_GROUND_SPEED_KT : (frame field) Ground Velocity, in knots.
- FM_AIRCRAFT_WINGSPAN_FT : (configuration field) Wingspan of the aircraft, in feet, measured in a straight line from wingtip to wingtip.
- FM_FLAP_LEFT_DEPLOYMENT : (frame field) Left Flap (trailing-edge) Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_FLAP_RIGHT_DEPLOYMENT : (frame field) Right Flap (trailing-edge) Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_SLAT_LEFT_DEPLOYMENT : (frame field) Left Slat (leading-edge flap) Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_SLAT_RIGHT_DEPLOYMENT : (frame field) Right Slat (leading-edge flap) Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_SPOILER_LEFT_DEPLOYMENT : (frame field) Left Spoiler Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_SPOILER_RIGHT_DEPLOYMENT : (frame field) Right Spoiler Deployment Level from 0.0 (fully retracted) to 1.0 (fully extended).
- FM_LANDING_GEAR_GENERAL_DEPLOYMENT : (frame field) General landing gear deployment level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR_ALL_FIXED : (configuration field) specifies if all landing gears are fixed.

Aircraft Detailed Landing Gears Information fields

- FM_LANDING_GEAR_COUNT : (configuration field) Landing gear count, up to 5.
- FM_LANDING_GEAR1_DEPLOYMENT : (frame field) Landing Gear 1 Deployment Level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR1_FIXED : (configuration field) specifies whether landing gear 1 is fixed.
- FM_LANDING_GEAR1_OFFSET_XYZ : (configuration field) Xyz vector of landing gear 1 position in meters, relative to [Reference].
- FM_LANDING_GEAR2_DEPLOYMENT : (frame field) Landing Gear 2 Deployment Level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR2_FIXED : (configuration field) specifies whether landing gear 2 is fixed.
- FM_LANDING_GEAR2_OFFSET_XYZ : (configuration field) Xyz vector of landing gear 2 position in meters, relative to [Reference].
- FM_LANDING_GEAR3_DEPLOYMENT : (frame field) Landing Gear 3 Deployment Level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR3_FIXED : (configuration field) specifies whether landing gear 3 is fixed.
- FM_LANDING_GEAR3_OFFSET_XYZ : (configuration field) Xyz vector of landing gear 3 position in meters, relative to [Reference].
- FM_LANDING_GEAR4_DEPLOYMENT : (frame field) Landing Gear 4 Deployment Level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR4_FIXED : (configuration field) specifies whether landing gear 4 is fixed.
- FM_LANDING_GEAR4_OFFSET_XYZ : (configuration field) Xyz vector of landing gear 4 position in meters, relative to [Reference].
- FM_LANDING_GEAR5_DEPLOYMENT : (frame field) Landing Gear 5 Deployment Level from 0.0 (fully retracted) to 1.0 (fully deployed).
- FM_LANDING_GEAR5_FIXED : (configuration field) specifies whether landing gear 5 is fixed.
- FM_LANDING_GEAR5_OFFSET_XYZ : (configuration field) Xyz vector of landing gear 5 position in meters, relative to [Reference].

Raw Motion Data fields (for advanced users only)

- FM_RAW_ROLL : (frame field) Left roll value transmitted directly to the Motion Output. Value should be between -1 and +1: -1 being max leaning to the right and +1 being max leaning to the left.
- FM_RAW_PITCH : (frame field) Forward pitch value transmitted directly to the Motion Output. Value should be between -1 and +1: -1 being max backward pitch and +1 being max forward pitch.
- FM_RAW_HEAVE : (frame field) Up heave value transmitted directly to the Motion Output. Value should be between -1 and +1: -1 being bottommost position and +1 being topmost position.

Custom states fields

Custom configuration fields when no generic meaning is suitable. When these are used, their actual meaning should be reported to D-BOX.

- FM_CONFIG_CUSTOM_01 : (configuration field)
- …
- FM_CONFIG_CUSTOM_10 : (configuration field)

Custom frame fields when no generic meaning is suitable. When these are used, their actual meaning should be reported to D-BOX.

- FM_FRAME_CUSTOM_01 : (frame field)
- …
- FM_FRAME_CUSTOM_10 : (frame field)

ACTION FIELDS

ACTION fields are used with ACTION events. The list built with REGISTER and UNREGISTER is global. The associations built with ASSIGN and UNASSIGN are also global.

- FM_ACTION : (Action field) unique numeric identification of action. Specific case for weapons: Each mode for a given weapon consists of a different resulting action therefore each of them requires a unique ACTION id that is usually computed by combining weapon id and mode id.
- FM_ACTION_STR : (Action field) unique string identification of action.
- FM_ACTION_SLOT : (Action field) Action Slot Index, usually 0-3 for 4 slots, 0-7 for 8 slots, etc. Action loaded in slot is ready to be used at any time (i.e. in-hand weapon).

EVENT FIELDS

EVENT FIELDS are posted when events occur and are immediately processed by the Motion Code event handler. Their state is only valid during the event local scope. No global state is maintained.

When a same EVENT FIELD is registered in different event structures, each posted event will be processed as distinct local values.

- FM_EVENT_INTENSITY : (Event field) Generic intensity attribute for events.
- FM_EVENT_YAW_RAD : (Event field) Generic direction.
- FM_EVENT_ORIENTATION_XYZ : (Event field) Generic orientation vector relative to body.
- FM_EVENT_POSITION_XYZ : (Event field) Generic position coordinates

Custom frame fields when no generic meaning is suitable. When these are used, their actual meaning should be reported to D-BOX.

- FM_EVENT_CUSTOM_01 : (Event field)
- …
- FM_EVENT_CUSTOM_10 : (Event field)

EVENT MEANINGS

This is the list of known meanings that can be associated to an event during registration. D-BOX may add meanings to this list over time but should not remove any. A same event meaning can be associated to multiple event keys.

- EM_UNKNOWN
- EM_CONFIG_UPDATE : Configuration update event (changing vehicle or context).
- EM_FRAME_UPDATE : Frame update event (generally at each simulation frame).
- EM_ENGINE_START : Vehicle engine started.
- EM_ENGINE_STOP : Vehicle engine stopped.
- EM_ENGINE_BOOST_START : Vehicle engine boost (or nitro) action started.
- EM_ENGINE_BOOST_STOP : Vehicle engine boost (or nitro) action ended.
- EM_IMPACT : General impact.
- EM_ACTION_REGISTER : Register an action to global list using ACTION field meanings. FM_ACTION (unique identifier) is mandatory. If unique identifier is already registered, it will be overwritten with new action.
- EM_ACTION_UNREGISTER : Unregister an action from global list. Only FM_ACTION (unique identifier) is mandatory.
- EM_ACTION_UNREGISTER_ALL : Unregister all actions from global list.
- EM_ACTION_ASSIGN : Assign an action to one of the 8 available slots, ready for use. FM_ACTION (unique identifier) must be specified along with FM_ACTION_SLOT to which action is to be assigned. If an action is already assigned to specified slot, assignment will be overwritten.
- EM_ACTION_UNASSIGN : Unassign an action from one of the 8 available slots. Only FM_ACTION_SLOT is mandatory.
- EM_ACTION_UNASSIGN_ALL : Unassign actions from all slots.
- EM_ACTION_TRIGGER_PULSE : Send an action trigger pulse (i.e. one-shot weapon fire).
- EM_ACTION_TRIGGER_START : Send an action trigger start (i.e. start of continuous weapon firing).
- EM_ACTION_TRIGGER_STOP : Send an action trigger stop (i.e. end of continuous weapon firing).

APPENDIX

Sample Racing delivered with D-BOX Live Motion SDK

```
#include <cstdio>
#include <conio.h>
#include "LiveMotionSdk/dboxLiveMotion.h"

#pragma comment(lib, DBOX_LIVEMOTION_LIB)

// TODO: Set APP_KEY to the key that was assigned to you by D-BOX.
#define APP_KEY                                 "SampleRacer"
// TODO: Set APP_BUILD to an integer representing your build number.
#define APP_BUILD                       1001

/// Sample configuration structure for CONFIG_UPDATE event.
// TODO: Unused fields can be removed from the structure
// and new relevant fields can be added to the structure.
// However the STRUCTINFO section must ALWAYS be matched.
struct RaceConfig {
        dbox::I32 EngineMaxRpm;
        dbox::I32 EngineRedRpm;
        dbox::I32 EngineIdleRpm;
        dbox::I32 EngineCylinders;
        dbox::F32 EngineMaxPower;
        dbox::F32 EngineMaxTorque;
        dbox::QuadFloat32 SuspensionMaxTravel;
        dbox::PCHAR VehicleTypeName;
        dbox::PWCHAR TrackName;

        DBOX_STRUCTINFO_BEGIN()
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineMaxRpm, dbox::FT_INT32, dbox::FM_ENGINE_RPM_MAX)
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineRedRpm, dbox::FT_INT32, dbox::FM_ENGINE_RPM_RED)
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineIdleRpm, dbox::FT_INT32, dbox::FM_ENGINE_RPM_IDLE)
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineCylinders, dbox::FT_INT32, dbox::FM_ENGINE_CYLINDERS)
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineMaxPower, dbox::FT_FLOAT32,
dbox::FM_ENGINE_POWER_MAX)
                DBOX_STRUCTINFO_FIELD(RaceConfig, EngineMaxTorque, dbox::FT_FLOAT32,
dbox::FM_ENGINE_TORQUE_MAX)
                DBOX_STRUCTINFO_FIELD(RaceConfig, SuspensionMaxTravel, dbox::FT_QUAD_FLOAT32,
dbox::FM_SUSPENSION_TRAVEL_MAX_QUAD)
                DBOX_STRUCTINFO_FIELD(RaceConfig, VehicleTypeName, dbox::FT_CHAR_PTR32,
dbox::FM_VEHICLE_TYPE_STR)
                DBOX_STRUCTINFO_FIELD(RaceConfig, TrackName, dbox::FT_WCHAR_PTR32, dbox::FM_MAP_STR)
        DBOX_STRUCTINFO_END()
};


/// Sample real-time frame structure for FRAME_UPDATE event.
// TODO: Unused fields can be removed from the structure
// and new relevant fields can be added to the structure.
// However the STRUCTINFO section must ALWAYS be matched.
struct RaceTelemetry {
        dbox::I32 EngineRpm;
        dbox::F32 EnginePower;
        dbox::F32 EngineTorque;
        dbox::I32 TransmissionGear;
        dbox::I32 TransmissionRpm;
        dbox::QuadFloat32 TireNormalizedLoad;
```

```
		dbox::QuadFloat32 TireSlipRatio;
		dbox::QuadFloat32 TireSlipAngleRadians;
		dbox::QuadFloat32 TireSlipVelocity;
		dbox::QuadFloat32 SuspensionTravel;
		dbox::XyzFloat32 Acceleration;
		dbox::XyzFloat32 Velocity;

		DBOX_STRUCTINFO_BEGIN()
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, EngineRpm, dbox::FT_INT32, dbox::FM_ENGINE_RPM)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, EnginePower, dbox::FT_FLOAT32, dbox::FM_ENGINE_POWER)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, EngineTorque, dbox::FT_FLOAT32, dbox::FM_ENGINE_TORQUE)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TransmissionGear, dbox::FT_INT32,
dbox::FM_ENGINE_TRANSMISSION_GEAR)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TransmissionRpm, dbox::FT_INT32,
dbox::FM_ENGINE_TRANSMISSION_RPM)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TireNormalizedLoad, dbox::FT_QUAD_FLOAT32,
dbox::FM_TIRE_LOAD_NORM_QUAD)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TireSlipRatio, dbox::FT_QUAD_FLOAT32,
dbox::FM_TIRE_SLIP_RATIO_QUAD)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TireSlipAngleRadians, dbox::FT_QUAD_FLOAT32,
dbox::FM_TIRE_SLIP_ANGLE_RAD_QUAD)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, TireSlipVelocity, dbox::FT_QUAD_FLOAT32,
dbox::FM_TIRE_SLIP_VELOCITY_QUAD)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, SuspensionTravel, dbox::FT_QUAD_FLOAT32,
dbox::FM_SUSPENSION_TRAVEL_QUAD)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, Acceleration, dbox::FT_XYZ_FLOAT32,
dbox::FM_ACCELERATION_XYZ)
			DBOX_STRUCTINFO_FIELD(RaceTelemetry, Velocity, dbox::FT_XYZ_FLOAT32, dbox::FM_VELOCITY_XYZ)
		DBOX_STRUCTINFO_END()
};

/// Sample attributes structure for events with level information.
struct LevelEvent {
		dbox::F32 Level;

		DBOX_STRUCTINFO_BEGIN()
			DBOX_STRUCTINFO_FIELD(LevelEvent, Level, dbox::FT_FLOAT32, dbox::FM_EVENT_INTENSITY)
		DBOX_STRUCTINFO_END()

		LevelEvent(dbox::F32 dLevel) :
			Level(dLevel) {
		}
};

/// Sample attributes structure for events with level and direction information.
struct DirLevelEvent {
		dbox::F32 Level;
		dbox::F32 Direction;

		DBOX_STRUCTINFO_BEGIN()
			DBOX_STRUCTINFO_FIELD(DirLevelEvent, Level, dbox::FT_FLOAT32, dbox::FM_EVENT_INTENSITY)
			DBOX_STRUCTINFO_FIELD(DirLevelEvent, Direction, dbox::FT_FLOAT32, dbox::FM_EVENT_YAW_RAD)
		DBOX_STRUCTINFO_END()

		DirLevelEvent(dbox::F32 dLevel, dbox::F32 dDirection) :
			Level(dLevel),
			Direction(dDirection) {
		}
};

/// These are your unique event ids that you'll use when calling PostEvent.
/// You can use the numbers that are the most appropriate for your application.
/// These values should be unique and not change in future releases.
enum AppEvents {
```

```
        RACE_CONFIG = 1000,
        RACE_TELEMETRY,
        ENGINE_STARTED = 2000,
        ENGINE_STOPPED,
        NITRO_STARTED,
        NITRO_FINISHED,
        CAR_BANG,
};

int main(int , char* []) {
        // Initialization and registration should be done only once.
        dbox::LiveMotion::Initialize(APP_KEY, APP_BUILD);
        dbox::LiveMotion::RegisterEvent(RACE_CONFIG, dbox::EM_CONFIG_UPDATE, RaceConfig::GetStructInfo());
        dbox::LiveMotion::RegisterEvent(RACE_TELEMETRY, dbox::EM_FRAME_UPDATE, RaceTelemetry::GetStructInfo());
        dbox::LiveMotion::RegisterEvent(ENGINE_STARTED, dbox::EM_ENGINE_START);
        dbox::LiveMotion::RegisterEvent(ENGINE_STOPPED, dbox::EM_ENGINE_STOP);
        dbox::LiveMotion::RegisterEvent(NITRO_STARTED, dbox::EM_ENGINE_BOOST_START, LevelEvent::GetStructInfo());
        dbox::LiveMotion::RegisterEvent(NITRO_FINISHED, dbox::EM_ENGINE_BOOST_STOP);
        dbox::LiveMotion::RegisterEvent(CAR_BANG, dbox::EM_IMPACT, DirLevelEvent::GetStructInfo());
        // Registration completed, open motion output device.
        // Open can be called once, after end of registration.
        dbox::LiveMotion::Open();
        {
                // Level Start
                // It is always good to ResetState before each race.
                dbox::LiveMotion::ResetState();
                RaceConfig oConfig;
                oConfig.EngineCylinders = 4;
                oConfig.EngineIdleRpm = 1100;
                oConfig.EngineMaxPower = 425 * 745.699872f;
                oConfig.EngineMaxRpm = 9000;
                oConfig.EngineMaxTorque = 312 * 1.35581795f;
                oConfig.EngineRedRpm = 7500;
                oConfig.SuspensionMaxTravel.BackLeft = 6 * 0.0254f;
                oConfig.SuspensionMaxTravel.BackRight = 6 * 0.0254f;
                oConfig.SuspensionMaxTravel.FrontLeft = 6 * 0.0254f;
                oConfig.SuspensionMaxTravel.FrontRight = 6 * 0.0254f;
                oConfig.TrackName = "Montreal Grand Prix";
                oConfig.VehicleTypeName = "GT";
                dbox::LiveMotion::PostEvent(RACE_CONFIG, oConfig);

                RaceTelemetry oTelemetry;
                oTelemetry.Acceleration.X = 0.0f;
                //oTelemetry...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);

                // Start of race, this will fade-in actual motion.
                dbox::LiveMotion::Start();
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                // Engine started, this is a sample event with no attributes.
                dbox::LiveMotion::PostEvent(ENGINE_STARTED);
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                // Nitro event, this is a sample event with level attribute.
                dbox::LiveMotion::PostEvent(NITRO_STARTED, LevelEvent(0.75f));          // 75% intensity
                // ...
                dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
```

```
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // Nitro finished, this is a sample event with no attributes.
                    dbox::LiveMotion::PostEvent(NITRO_FINISHED);
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // Impact, this is a sample event with direction and level attributes.
                    dbox::LiveMotion::PostEvent(CAR_BANG, DirLevelEvent(0.4f, 1.57f));          // 40% intensity, 90 degrees.
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // Pause, this will fade-out motion.
                    dbox::LiveMotion::Stop();

                    // Resume from pause, this will fade-in actual motion.
                    dbox::LiveMotion::Start();
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // Engine cut off, this is a sample event with no attributes.
                    dbox::LiveMotion::PostEvent(ENGINE_STOPPED);
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // ...
                    dbox::LiveMotion::PostEvent(RACE_TELEMETRY, oTelemetry);
                    // End of level, this will fade-out motion.
                    dbox::LiveMotion::Stop();

                    // Level End
            }
            // Close motion output device.
            dbox::LiveMotion::Close();
            // Terminate
            dbox::LiveMotion::Terminate();

            printf("\nEnded, press any key...\n");
            _getch();

            return 0;
}
```