Telemetry output of IL2 v4.003

revision 1.2

1. Basic setup

Starting from 4.003 version IL2 Grate Battles supports Telemetry output, allowing connecting telemetry indicators device (simultaneously one connected device) via UDP transport layer with specified IP address and port.

Basic setup of output is done in "data/startup.cfg", section "TelemetryDevice". Default parameters are:

```
[KEY = telemetrydevice]
     addr = "127.0.0.1"  //target address 1
     addr1 = "127.0.0.1:1001" //target address 2
     decimation = 2      //decimation rate of 50Hz output
     enable = false      //true to enable output
     port = 4322         //target port
[END]
```

addr parameter specifies destination address in form "IP[:port]". If port number is not specified explicitly then *port* value will be used for sending data to this IP

It is possible to specify several a*ddr* parameters to send information to different target addresses in form *addrN* (addr0="x.x.x.x", addr1="x.x.x.x" etc.)

Simulation produces 50Hz rate data output (output 50 samples per second). To reduce UDP messages output rate the above setup section contains an integer setting "decimation":

UDP_output_rate = Data_output_rate / decimation

The default setup makes UDP output rate half of simulation's rate and is equal 25Hz.

2. Output state

Simulation system produces permanent plane/vehicle indicators messages and single event messages for in-game events.

When user activates input-focus on the controlled object (such as Plane or Turret) system sends to telemetry device an event called SET_FOCUS which describes corresponding controlled object's name. Usually SET_FOCUS is followed by an additional SETUP_x events which provide more initialization information for controlled object's systems

Permanent telemetry data (such as Engine RPM etc.) is delivered through SIndicator structure constantly to telemetry device

In-game single events such as bullet-hit, gun fire etc. are sent through single-shoot events and described in SEvent structure

3. UDP message

The format of UDP message is shown below

Member	**Offset**	**Size**	**Type**	**Comment**
Packet_ID	0	4	DWORD	Valid value: `0x54000101`
Size	4	2	WORD	Current message size, bytes
Tick	6	4	DWORD	Simulation frame tick (1 unit = 1/50 second)
Indicators_Count	10	1	BYTE	Total number of indicators below (IC)
Indicator[0]	11		SIndicator	
Indicator[...]			SIndicator	
Indicator[IC-1]			SIndicator	
Events_Count		1	BYTE	Total number of events (EC)
Event[0]			SEvent	
Event[...]			SEvent	
Event[EC-1]			SEvent	

Type *float* corresponds to *float IEEE 754* floating point type;

Type *DWORD* corresponds to LSB unsigned integer (4 bytes)

Type *WORD* corresponds to LSB unsigned short integer (2 bytes)

Type *BYTE* corresponds to LSB unsigned char value (1 byte)

Type *STRING* consists of sequence: String Length (1 byte), following string ASCII data

3.1. SIndicator

Indicator structure. Describes permanent indicator state:

Member	**Offset**	**Size**	**Type**	**Comment**
Indicator_ID	0	2	WORD	
Values_count	2	1	BYTE	Total number of Indicator's values below (VC)
Indicators[0]	3	4	32-bit	Indicator data
Indicator[..]		4	32-bit	
Indicators[VC-1]		4	32-bit	

3.2. Indicators list

Currently given set of indicators are supported:

Name	**ID**	**Comment**
ENG_RPM	0	Engines RPM. *Values_count* of SIndicator structure equals to currently installed engines. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single engine's RPM
ENG_MP	1	Engines manifold pressures. *Values_count* of SIndicator structure equals to currently installed engines. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single engine's Manifold Pressure
ENG_SHAKE_FRQ	2	Engines shake frequency. *Values_count* of SIndicator structure equals to currently installed engines. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single engine's Shake Frequency (Hz)
ENG_SHAKE_AMP	3	Engines shake amplitude. *Values_count* of SIndicator structure equals to currently installed engines. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single engine's Shake Amplitude [0..1]
LGEARS_STATE	4	Landing gears extraction state. *Values_count* of SIndicator structure equals to currently available landing gears. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single landing gear Extraction State [0..1]
LGEARS_PRESS	5	Landing gears pressure state. *Values_count* of SIndicator structure equals to currently available landing gears. Each indicator value of SIndicator structure should be interpreted as 32-bit *float* value which refers to single landing gear Pressure Magnitude [0..1]
EAS	6	Single indicator value that corresponds to 32-bit *float* EAS value (m/s)
AOA	7	Single indicator value that corresponds to 32-bit *float* AOA value (rad)
ACCELERATION	8	Three-component 32-bit *float* vector that describes Acceleration vector including Gravity
COCKPIT_SHAKE	9	Two-component 32-bit *float* vector that describes cockpit shake Frequency (Hz) and Amplitude [0..1] values
AGL	10	Single indicator value that corresponds to 32-bit *float* Altitude above Ground Level (m)
FLAPS	11	Flaps extraction state 32-bit *float* value [0..1]
AIR_BRAKES	12	Air brakes extraction state 32-bit *float* value [0..1]

3.3. SEvent

Event structure. Describes single event data:

Member	**Offset**	**Size**	**Type**	**Comment**
Event_ID	0	2	WORD	
Event_Size	2	1	BYTE	Size of event's data
Data...	3	…		Event's data

3.4. Events list

Most of events are represented as data structures packet with 2-bytes packing alignment. Currently given set of events are supported:

Name	**ID**	**Comment**
SET_FOCUS	0	Object's name (String) on which user input focus was applied.

SETUP_ENG	1	Event's data corresponds to following data structure
		`struct STEEngineSetup{`
		`        short nIndex;           // 0,1,2,...`
		`        short nID;`
		`        float afPos[3];         // Local engine position, [m]`
		`        float fMaxRPM;          // [rev/min], (>0)`
		`        };`

SETUP_GUN	2	Event's data corresponds to following data structure
		`struct STEGunSetup{`
		`        short nIndex;           // 0,1,2,...`
		`        float afPos[3];         // Local gun position, [m]`
		`        float fProjectileMass;  // [kg]`
		`        float fShootVelocity;   // [m/s]`
		`        };`

SETUP_LGEAR	3	Event's data corresponds to following data structure
		`struct STELandingGearSetup{`
		`        short nIndex;           // 0,1,2,...`
		`        short nID;`
		`        float afPos[3];         // Local landing-gear position, [m]`
		`        };`

DROP_BOMB	4	Event's data corresponds to following data structure
		`struct STEDropData{`
		`        float afPos[3];         // Local coords of drop position, [m]`
		`        float fMass;            // Released mass [kg]`
		`        unsigned short uFlags;  // Bit 1 = Rocket`
		`        };`

ROCKET_LAUNCH	5	Event's data same as DROP_BOMB

HIT	6	Event's data corresponds to following data structure
		`struct STEHit{`
		`        float afPos[3];         // Local coords of hit position, [m]`
		`        float afHitF[3];        // Local hit force, [N]`
		`        };`

DAMAGE	7	Event's data same as HIT

EXPLOSION	8	Event's data corresponds to following data structure
		`struct STEExplosion{`
		`        float afPos[3];         // Local coords of explosion position, [m]`
		`        float fExpRad;          // Explosion radius, [m]`
		`        };`

GUN_FIRE	9	Event's data contains single *byte* corresponding to index of firing gun as single bullet event

SRV_ADDR	10	Game server “HostAddress:port” (String), available in network gaming
SRV_TITLE	11	Game server Title (String), available in network gaming
SRS_ADDR	12	SRS server “HostAddress:port” (String), available in network gaming
CLIENT_DAT	13	User's client data structure. Sent once after entering server `struct STClientData{` `long nClientID,` `//User's ClientID` `nServerClientID;` `//Server's ClientID` `char sPlayerName[32];` `};`
CTRL_DAT	14	Controlled object data. Sent on object creation and parent ID changes `struct STControlledData{` `long nParentClientID; //ClientID of parent vehicle (if has, else -1)` `short nCoalitionID;` `};`

Revision history

1.1. Added members *fMass*, *uFlags* in **STEDropData** structure for DROP_BOMB event. Added member *nID* in **STEEngineSetup** and **STELandingGearSetup** structures. Default UDP port is 4322

1.2. Changes in section “1. Basic setup”, several target addresses are now supported. New events were introduced: SRV_ADDR, SRV_TITLE, SRS_ADDR, CLIENT_DAT, CTRL_DAT