AC Remote Telemetry Documentation

This document helps you to set up UDP socket comunication between your application and Assetto Corsa. All the code samples are made using http://codepad.org/ .

All the code snippets follow C++ syntax. The PC running Assetto Corsa will be referred as the ACServer.
int are 32 bit little endian integers

float are 32 bit floating point numbers

bool are 8 bit boolean value

1) Connect to AC via UDP for Remote Telemetry : Handshake

1.1) Client starts comunication with AC Remote Telemetry Server

The handshake process to connect and receive Remote Telemtry data from Assetto Corsa via UDP works as follows: your app first must create an UDP socket and connect to a PC address running Assetto Corsa (ACServer). The connection port number is

9996

Your application must send a structured data using the following format :

```
struct handshaker{

	int identifier;
	int version;
	int operationId;
};
```

3 integers are required for the handshaking phase:

- int identifier : [not used in the current Remote Telemtry version by AC] In future versions it will identify the platform type of the client. This will be used to adjust a specific behaviour for each platform :

  eIPhoneDevice=0

  eIPadDevice=1

  eAndroidPhone=2

  eAndroidTablet=3

- int version : [not used in the current Remote Telemtry version by AC] In future version this field will identify the AC Remote Telemetry version that the device expects to speak with.

- int operationId: This is the type of operation required by the client. The following operations are now available:

- HANDSHAKE = 0 :

This operation identifier must be set when the client wants to start the comunication.

- SUBSCRIBE_UPDATE = 1 :

This operation identifier must be set when the client wants to be updated from the specific ACServer.

- SUBSCRIBE_SPOT = 2 :

This operation identifier must be set when the client wants to be updated from the specific ACServer just for SPOT Events (e.g.: the end of a lap).

- DISMISS = 3 :

This operation identifier must be set when the client wants to leave the comunication with ACServer.

In summary, for the first handshaking phase your application will need to send the following structured data to ACServer

```
struct handshaker;
handshaker.identifier = 1 ;
handshaker.version = 1 ;
handshaker.operationId= 0 ;
```

1.2) AC Remote Telemetry Server Responds to the client

After sending the structured data in section 1.1. your application will receive the following struct as response

```
struct handshackerResponse{
	char carName[50];
	char driverName[50];
	int identifier;
	int version;

	char trackName[50];

	char trackConfig[50];
};
```

- char carName[50] : is the name of the car that the player is driving on the AC Server
- char driverName[50] :is the name of the driver running on the AC Server
- int identifier : for now is just 4242, this code will identify different status, as “NOT AVAILABLE” for connection
- int version : for now is set to 1, this will identify the version running on the AC Server
- char trackName[50] :is the name of the track on the AC Server
- char trackConfig[50]: is the track configuration

Your application will need to parse this structured data in order to get the information.

This step is necessary to understand which driver are we connecting to.

1.3) AC Client confirms connection

Again the client must send the following structured data, the same from section 1.1 :

```
struct handshaker{

	int identifier;
	int version;
	int operationId;
};
```

Now operationId must be one of the following options :

- SUBSCRIBE_UPDATE = 1 :

This operation identifier must be set when the client wants to be updated from the specific ACServer.

- SUBSCRIBE_SPOT = 2 :

This operation identifier must be set when the client wants to be updated from the specific ACServer just for SPOT Events (e.g.: the end of a lap).

After this phase the Client is added as a listener to AC Remote Telemetry listeners.

2) ACServer updating clients

For each physics step, ACServer will call the update function to all the listeners.

- If the client subscribed himself with SUBSCRIBE_UPDATE identifier, it will receive the following structured data

```
struct RTCarInfo
{
	char identifier;
```

```
        int size;

        float speed_Kmh;
        float speed_Mph;
        float speed_Ms;

        bool isAbsEnabled;
        bool isAbsInAction;
        bool isTcInAction;
        bool isTcEnabled;
        bool isInPit;
        bool isEngineLimiterOn;


        float accG_vertical;
        float accG_horizontal;
        float accG_frontal;

        int lapTime;
        int lastLap;
        int bestLap;
        int lapCount;

        float gas;
        float brake;
        float clutch;
        float engineRPM;
        float steer;
        int gear;
        float cgHeight;

        float wheelAngularSpeed[4];
        float slipAngle[4];
        float slipAngle_ContactPatch[4];
        float slipRatio[4];
        float tyreSlip[4];
        float ndSlip[4];
        float load[4];
        float Dy[4];
        float Mz[4];
        float tyreDirtyLevel[4];

        float camberRAD[4];
        float tyreRadius[4];
        float tyreLoadedRadius[4];

    float suspensionHeight[4];

        float carPositionNormalized;
```

```
    float carSlope;

float carCoordinates[3];

} ;
```

- char identifier : is set to char “a” , it is used to understand that the structured data is the data that the client app wants
- int size : the size of the structured data in Bytes.

- If the client subscribed himself with SUBSCRIBE_SPOT identifier, it will receive the following structured data whenever a spot event is triggered (for example for the end of a lap). Differently from SUBSCRIBE_UPDATE , this event will interest all the cars in the AC session:

```
struct RTLap
{
    int carIdentifierNumber;
    int lap;
    char driverName[50];
    char carName[50];
    int time;
};
```

Your application will need to parse locally the structured data sent by ACServer

3) Dismissing an AC Client

A client to dimiss himself he must send the following package (the same from session 1.1)

```
struct handshaker{

    int identifier;
    int version;
    int operationId;
};
```

with DISMISS = 3 as operationId.

The client will be removed from the listeners, ACServer will forget about him and no more updates will be sent to him. To connect again, do again the steps from section 1.1 or 1.2