

*CYBERNET*

**OpenSkies**

Networking Engine

## Database Connection Utility Guide



[www.openskies.net](http://www.openskies.net)

**MMPG**  
**MASSIVE**  
**MULTIPLAYER**  
**ONLINE GAMING**

## DB\_Connection Guide

This document provides a brief description of the functionality provided by the CDB\_Connection class which is provided in the Openskies SDK as source. These two files are located in:

```
\CybernetRTI_Static\CybernetRTI_SDK\include\DB_Connection.h  
\CybernetRTI_Static\CybernetRTI_SDK\src\DB_Connection.cpp
```

Incorporating these into your application requires the OpenSSL libraries and headers which can be downloaded from [www.OpenSSL.org](http://www.OpenSSL.org).

### Purpose

The CDB\_Connection class provides a secure communication channel to the Openskies SQL Database Server, which resides on a Linux/Netmax machine. It is set up to communicate user information. This user information corresponds to the fields in the aforementioned SQL database. The Openskies DB Server runs a cgi script whenever a communication is received from a client. This script decrypts and parses the information sent in the CDB\_Connection code, access the SQL database, and then responds to the communication by sending information back to the client. The user fields are:

- Userid
- Password
- Cookie – a 32 byte encryption key generated on the Openskies DB Server
- Lat – initial player location generated on the Openskies DB Server
- Lon – initial player location generated on the Openskies DB Server
- Kills – how many opponents the player has destroyed
- Deaths – how many times the player has been destroyed
- Score – player's overall score

The simple API in the CDB\_Connection class allows the client to request and update these fields.

### Functions

The following is a brief description of the functions in the CDB\_Connection class.

- **CDB\_Connection(void)** - constructor, initializes some member variables, does not initiate connection.
  - No arguments
- **~CDB\_Connection(void)** - destructor
  - No arguments

- **bool Initialize(const char \*address, DWORD port, const char \*userid, const char \*password)** – sets the CDB\_Connection member variables corresponding to those passed in. This function does not initiate connection.
  - *Address* (in) – a null-terminated string that is the IP address of the Openskies DB Server.
  - *Port* (in) – This is the Openskies DB Server port that is listening for communication (see the **Server DB Guide**).
  - *UserId* (in) – a null-terminated string that specifies this player's unique ID.
  - *Password* (in) – this player's password
- **bool Login(char \*cookie, float &lat, float &lon, int &kills, int &deaths, int &score)** – This function initiates a connection with the Openskies DB Server and at the same time receives all of its DB information. *Initialize* must be called *before* this function is called. Returns true if connection was successful, false otherwise.
  - *Cookie* (Out) – this is a 32-byte encryption key generated on the Openskies DB Server and is returned to the client via this function. This cookie is then passed into *CNetInterface* functions (See *Openskies IET* documentation) like *JoinScenario* to authenticate this user to the federation.
  - *Lat* (Out) – this is half of the initial location for this player. It is generated on the Openskies DB server and sent to the player via this function.
  - *Lon* (Out) – this is half of the initial location for this player. It is generated on the Openskies DB server and sent to the player via this function.
  - *Kills* (Out) – the total number of kills for this player is stored on the Openskies Server and is returned via this function.
  - *Deaths* (Out) – the total number of times this player has been destroyed is stored on the Openskies Server and is returned via this function.
  - *Score* (Out) – the total score for this player is stored on the Openskies Server and is returned via this function.
- **bool Die(int &kills, int &deaths, int &score)** – this function commands the Openskies DB Server to increment the Deaths for this player by one. It also returns some player state via the arguments. Returns true if connection was successful, false otherwise.
  - *Kills* (Out) – the total number of kills for this player is stored on the Openskies Server and is returned via this function.
  - *Deaths* (Out) – the total number of times this player has been destroyed is stored on the Openskies Server and is returned via this function.
  - *Score* (Out) – the total score for this player is stored on the Openskies Server and is returned via this function.
- **bool Kill(void)** – this function commands the Openskies DB Server to increment the Kill for this player by one. Returns true if connection was successful, false otherwise.
  - No arguments
- **bool KillTarget(int &kills, int &deaths, int &score)** – This function updates the number of target kills (where the player has killed his

assigned target). Target kills are worth more points (score) than other kills. Returns true if connection was successful, false otherwise.

- *Kills* (Out) – the total number of kills for this player is stored on the Openskies Server and is returned via this function.
- *Deaths* (Out) – the total number of times this player has been destroyed is stored on the Openskies Server and is returned via this function.
- *Score* (Out) – the total score for this player is stored on the Openskies Server and is returned via this function.
- **bool Logout(void)**- Informs the Openskies DB Server that we are quitting. Returns true if connection was successful, false otherwise.
- **bool Heartbeat(void)**- this function informs the Openskies DB Server that this player is still connected. This is primarily for player status on the Openskies DB server. The idea was to call this function once every 5 minutes or so. Returns true if connection was successful, false otherwise.
- **bool CheckVersion(HMODULE hInstance)**- This function determines if this players version of the game/applicaton is up-to-date. Returns true if connection was successful and the version is current, otherwise returns false.
  - **hInstance** – this is the module Handle for this application. (CWinApp.m\_hInstance). It allows this function to get the version numbers from the resource. In the end, this function fills an 8-byte with this information which it sends to the Openskies DB Server. Feel free to modify this code to fill that version buffer your own way.

o