|  |
| --- |
| TheNoise |
| Client-Server Protocol and Database Structure |
| An Overview |

|  |
| --- |
| Thomas Bloom, Matthew Lindsay  4/17/2013 |

**Abstract:** This document provides insight into the low level data communication and high level application control protocol used in the TheNoise audio streaming system.

**Protocol Definition:** The required functionality of TheNoise (See: Design Overview) dictates the communication protocol must support user authentication, push notifications to the client, and audio streaming to the client. A general design guideline during the development of this protocol was to be as flexible as possible, allowing introduction of future features. To enforce this flexibility the protocol was segmented into two separate major layers; the Low Level Protocol (LLP) and High Level Protocol (HLP).

**LLP Definition:** The low level protocol operates directly on top of a TCP connection. TCP is very reliable but is inherently a stream based protocol. Due to input requirements of the HLP, the LLP must encapsulate this stream nature and deliver atomic chunks of data to anything above it. Additionally for efficiency the LLP will provide a simple means of identifying the payload of a packet.

**Atomicity:** Packet atomicity ensures higher level layers do not receive partial payloads. If received data can always be assumed complete, higher levels become much simpler to implement and analyze. The LLP ensures atomicity of each payload by reading the packet length from every incoming header, and buffering the entire payload content before notifying higher levels of reception. Since TCP ensures reliable delivery and ordering, this technique guarantees complete payload atomicity.

**Payload Identification:** The packet header contains a Packet Type field of one byte. This field is not used directly by the LLP, freeing a simple means of fast payload identification for higher layers. An example of use would be to give authentication packets one payload identifier (PI), while audio stream packets get another PI. This simple method of identification allows higher levels to immediately begin processing packet payloads based on known structure rather than needing to universally parse each one.

|  |  |
| --- | --- |
| **Byte** | **0** |
| **0** | SOH (0x01) |
| **1** | Packet Type |
| **2** | Length High Order |
| **3** | Length Low Order |
| **4-65539** | Payload (optional) |

Table 1 – Low Level Packet for TheNoise Client/Server Protocol

**Protocol Implementation:** The LLP is completely implemented inside of

**HLP Definition:** The High Level Protocol builds on the output of the LLP. Since the output of the LLP is strict, the HLP concentrates only on flexibility. To this end, most HLP content is serialized using the .net framework DataContractSerializer. Direct serialization of objects is possible due to the Packet Type identifier in the LLP. The packet type determines what type of object is contained in the payload, allowing direct deserialization at the receiving end.

**Packet Types:** There are currently seven packet types. A table of these types is below as Table 2. The blue colored payloads are serialized objects of the same name.

|  |  |  |  |
| --- | --- | --- | --- |
| **Packet Name** | **Type ID** | **Payload Client->Server** | **Payload Server->Client** |
| Authenticate | 0 | LoginData | UserAuthenticationResult |
| Register | 2 | LoginData | UserAddResult |
| RequestList | 4 | None | TrackList |
| StartAudioStream | 5 | TrackStreamRequest | Does not send |
| StopAudioStream | 6 | None | Does not send |
| AudioSegment | 7 | Does not send | Binary stream of audio |
| UnknownPacket | 90 | Unused | Unused |

Table 2 – List of Packet Types

**Database Design:**