RFC 001: R-Type Game Protocol (RTGP)

Version: 1.0.0
Status: Draft / Experimental
Date: 2025-11-25
Authors: R-Type Project Team

Abstract

This document specifies the binary application-layer protocol used for real-time communication between the R-Type Client and Server.

1. Introduction

The R-Type Game Protocol (RTGP) is a lightweight, binary, datagram-oriented protocol designed to facilitate real-time multiplayer gameplay.
It prioritizes low latency and bandwidth efficiency. This specification provides sufficient detail for a third-party developer to implement
a compatible Client or Server without access to the reference implementation source code.

2. Terminology & Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL"
in this document are to be interpreted as described in RFC 2119.

2.1 Data Types

Byte: 8-bit unsigned integer.
uint16: 16-bit unsigned integer.
uint32: 32-bit unsigned integer.
float: 32-bit IEEE 754 floating point.
String: NOT SUPPORTED in standard packets to avoid allocation overhead, unless specified in the payload.

2.2 Byte Order

All multi-byte numeric fields MUST be transmitted in Network Byte Order (Big-Endian).
Implementations on Little-Endian architectures (x86/x64) MUST convert data before transmission (htons, htonl) and after reception (ntohs, ntohl).

3. Transport Layer

Protocol: UDP (User Datagram Protocol)
Default Port: 4242
MTU Safety: The total packet size (Header + Payload) SHOULD NOT exceed 1400 bytes to avoid IP fragmentation on standard networks.

4. Packet Structure

Every RTGP packet consists of a fixed 8-byte Header followed by a variable-length Data Payload.

4.1 Header Format

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Magic Byte   |  Packet Type  |          Packet Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            User ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                       Data Payload ...                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field Definitions:

Magic Byte (uint8): MUST be 0xA1.

Packet Type (uint8): The Operation Code (OpCode) defined in Section 5.

Packet Size (uint16): Length of the Data Payload in bytes.

User ID (uint32): The sender's unique identifier.

4.2 User ID Conventions

Server Authority: 0xFFFFFFFF (Integer -1)

Unassigned Client: 0x00000000

Assigned Client: 0x00000001 to 0xFFFFFFFE

5. Protocol Operations (OpCodes)

5.1 Session Management

0x01 - C_CONNECT: Client requests connection.

0x02 - S_ACCEPT: Server accepts connection and assigns User ID.

0x03 - DISCONNECT: Client or Server terminates session.

0x04 - C_GET_USERS: Client requests list of connected players.

0x05 - S_UPDATE_STATE: Server notifies clients of global game state.

5.2 Gameplay & Entity Management

0x10 - S_ENTITY_SPAWN: Server instructs clients to instantiate a game object.

0x11 - S_ENTITY_MOVE: Server updates remote entities' positions.

0x12 - S_ENTITY_DESTROY: Server instructs clients to remove an entity.

5.3 Input & Reconciliation

0x20 - C_INPUT: Client sends current input state.

0x21 - S_UPDATE_POS: Server sends authoritative correction for client position.

6. Security Considerations

Header Validation: Drop packets where Header[0] != 0xA1.

Spoofing Protection: Server must verify User ID matches sender's IP/Port.

Authority Check: Clients ignore packets claiming 0xFFFFFFFF unless from known Server IP.

7. Future Extensions

Reserved OpCodes 0xF0 to 0xFF are set aside for debugging and ping measurements.