Simple Query and Response Protocol (SQRP) Version 1.1

Abstract

This document outlines the Simple Query and Response Protocol (SQPR) for handling queries and responses between a client and server. It includes details on facilitating checks for directory and file existence, file modification status, and identification of modified files with a specified extension. This document specifies the SQRP/1.1's transport layer protocol.

Table of Contents

Abstract 2
1. Introduction 3
2. Protocol Overview 3
3. Message Format 4
  3.1. Header Format 4
  3.2. Body Format 5
4. Procedures 6
  4.1. Initialization Procedure 6
  4.2. Communication Procedure 7
Appendix A. Changes from Previous RFCs 7
  A.1 Changes from SQRP/1.0 7

1. Introduction

The Simple Query and Response Protocol (SQRP) aims to provide a structured framework for managing queries and responses between a client and server. SQRP Version 1.1 is specifically designed for checking files on the server. It offers a streamlined approach for handling tasks such as verifying directory and file existence, determining file modification status, and identifying modified files with specified extensions.

This document outlines the key components of SQRP. It includes its protocol overview, message format, procedures, and communication protocols. SQRP aims to facilitate efficient communication and data retrieval in a standardized manner by offering a comprehensive solution to these common server-client interactions.

The SQRP Version 1.1 operates on a client-server model and uses an 8-byte header structure. The header includes important fields such as Message Type, Query Type, Message ID, Time Stamp, Status Code, Body Length, and Reserved for future enhancements. SQRP supports four distinct operations, allowing users to verify directory existence, check file existence, determine file modification status, and identify modified files with specified extensions.

The following sections of this document discuss the protocol's details, including message format, header structure, body format, initialization procedures, and communication protocols. SQRP prioritizes efficiency, clarity, and flexibility in its design, providing a robust solution for handling various query and response scenarios.

SQRP is a dynamic and adaptable protocol that simplifies interactions between clients and servers, providing a reliable foundation for querying and retrieving information. The following sections provide a comprehensive guide to SQRP's structure, enabling users to implement and leverage its capabilities effectively.

2. Protocol Overview

SQRP version 1.1 operates on a client-server model and is built around an 8-byte header structure. The header includes important fields such as Message Type, Query Type, Message ID, Time Stamp, Status Code, Body Length, and Reserved for future enhancements. The protocol supports four different operations: verifying directory existence, verifying file existence, determining file modification status, and identifying modified files with a given extension. SQRP/1.1 communication generally takes place over TCP/IP connections. The default port is TCP-31369, other ports can be used. This does not preclude SQRP from being implemented on top of any other protocol on the Internet, or on other networks.

3. Message Format

SQRP needs an 8-byte header and maximum 256-bytes body due to the Body Length.

3.1. Header Format

7	6	5	4	3	2	1	0	Bits
Message	**Query Type**		**Message ID**					**7**
Time Stamp								**6**
								5
								4
								3
Status Code				**Reserved**				**2**
Body Length								**1**
Reserved								**0**

Figure1 General Representation of Header Format

Message Type (1-bit): Defines the type of the header it is in bit 63.

- **0b = QUERY**
- **1b = RESPONSE**

Query Type (2-bits): Identifies the operation. Bit 62-61

- **00b = Verify directory existence**
- **01b = Check file existence**
- **10b = Determine if an existing file has been modified after a specified timestamp**
- **11b = Identify files with a specified extension that have been modified after a given timestamp**

Message ID (5-bits): Provides a unique identifier for tracking the message. Bit 60-56

- **Encoding is converting binary form into integer between 31-0**
- **Ex. 10011b = 19**

Time Stamp (32-bits): Embeds a compact representation of the "modified after" time. Bit 55-24

- **55-51 Hour encoding is converting binary form into integer between 23-01**
  - **Ex. 10101b = 21**
- **50-45 Minute encoding is converting binary form into integer between 59-00**
  - **Ex. 111011b = 59**
- **44-39 Second encoding is converting binary form into integer between 59-00**
  - **Ex. 111011b = 59**
- **38-34 Day encoding is converting binary form into integer between 31-01**
  - **Ex. 10011b = 19**
- **33-30 Month encoding is converting binary form into integer between 12-01**
  - **Ex. 1100b = 12**

- **29-24 Year encoding is taking base as 2020 which means 0 in our case then increasing 0 means increasing 2020. 6 bit means 63 is the max value so the protocol can exist until 2020 + 63 = 2083**
    - **Ex. 110100b = 52 which means 2020 + 52 = 2072**

Status Code (3-bits): Indicates the outcome of the operation. Bit 23-21

- **000b = EXIST**
- **001b = NOT EXIST**
- **010b = CHANGED**
- **011b = NOT CHANGED**
- **100b = BAD REQUEST**
- **101b = ID LEASED**
- **110b = DIRECTORY NEEDED**
- **111b = SUCCESS**

Reserved (5-bits): Bit 20-16

Body Length (8-bits): Defines the coming message body's length. Bit 15-8

- **Encoding is converting binary to integer in terms of bytes**
    - **Ex. 11100111b = 231 bytes**
    - **This means that body can be maximum of 231 bytes for the example**

Reserved (8-bits): Bit 7-0

3.2. Body Format

Body Format is in extended ASCII format. Each object separated with comma (,)

- **Ex. 01101001(i) 01110011(s) 01101001(i) 01101011(k) 00101110(.) 01101010(j) 01110000(p) 01100111(g) 00101100(,) 01110100(t) 01100101(e) 01110011(s) 01110100(t) 00101110(.) 01100101(e) 01111000(x) 01100101(e) = isik.jpg,test.exe which is 17 bytes in total**

4. Procedures

Verify directory existence: Takes the given directory from related query's message body. After checking the existence of the directory, it responds with the result through response message if needed server can give the related information with message body.

Check file existence: Takes the given directory from previous directory existence query and takes the given file from related query's message body. After checking the existence of the file in the related directory, it responds with the result through response message if needed server can give the related information with message body.

Determine if an existing file has been modified after a specified timestamp: Takes the given directory from previous directory existence query and takes the given file from related query's message body and takes the timestamp from message header. After checking the existing file's last modified time the server response, the related message from message header and body.

Identify files with a specified extension that have been modified after a given timestamp: Takes the given directory from previous directory existence query and takes the given file extension from related query's message body. After checking the directory for the files with the related extension and checking the files' last modified times. Server responds with the related message from message header and body.

4.1. Initialization Procedure

1 = Verify directory existence

2 = Check file existence

3 = Determine if an existing file has been modified after a specified timestamp

4 = Identify files with a specified extension that have been modified after a given timestamp

In order to "1" happen query message should send with "00" query type and directory name encoded body should send with correct body length. If a directory exists, the RESPONSE message should have "000" EXIST status code. Else the RESPONSE message should have "001" NOT EXIST status code.

In order to "2" happen "1" should happened first with "000" EXIST status code. If not, the RESPONSE message returns with "110" DIRECTORY NEEDED status code. After that query message should send with "01" query type and file name encoded body should send with correct body length. If a file exists, the RESPONSE message should have "000" EXIST status code. Else the RESPONSE message should have "001" NOT EXIST status code.

In order to "3" happen "2" should happened first with "000" EXIST status code. If not, the RESPONSE message returns with "001" NOT EXIST status code this means file. After that query message should send with "10" query type, related timestamp and file name encoded body should send with correct body length. If a file changed, the RESPONSE message should have "010"

CHANGED status code with timestamp in the message body. Else the RESPONSE message should have "011" NOT CHANGED status code with last changed info send with the message body.

In order to "4" happen "1" should happened first with "000" EXIST status code. If not, the RESPONSE message returns with "110" DIRECTORY NEEDED status code. After that query message should send with "11" query type, related timestamp and extension name encoded body should send with correct body length. If a file exists and changed after the given timestamp, the RESPONSE message should have "111" SUCCESS status code with related filenames and timestamps encoded message body. Else the RESPONSE message should have "001" NOT EXIST status code.

4.2. Communication Procedure

Bad request: If the header format is not in the right condition the RESPONSE returns with "100" BAD REQUEST status code.

Leasing message ID: Every client needs a unique message ID to communicate with the server. There are 32 of them 0-31. Once the query is sent with a message ID between 0-31 if the response is not "100" BAD REQUEST the related message ID leased to the related client. If client is using other clients leased message ID RESPONSE returns with "101" ID LEASED status code. Leasing message ID ends after 5 minutes of no request.

Appendix A. Changes from Previous RFCs

A.1 Changes from SQRP/1.0

Since SQRP/1.0 did not give the transport layer protocol and its port there can be unclarity for the protocol users. Related changes are made now SQRP uses TCP/IP for transport and network layer, and port 31369 used in default.