RFC-6377.

Definition of Arithmetic Operations Transfer Protocol (AOTP)

Current Edition: Protocol Version 1.2

For Network & Internet Planning Consortium
Of 12 of the World's Top Networking Scientists

Guided by Dr. Omprakash Gnawali, Genius Extraordinaire

Collaboration & Planning Committee:

Wellington Cabrera, Ph.D., Founder – Version 1.0 – February 4, 2015

***Professor Prudhvi*, Research Scientist – Version 1.1 – February 4, 2015**

Mr. David Fullerton, Maverick Entrepreneur – Version 1.2 – February 5, 2015

Dr. David Matusevich, Regular Guy – Version 1.3 - February 5, 2015

Version History		
Version	**Author**	**Changes**
1.0	**Wellington Cabrera**	**Initial Release**
1.1	**Prudhvi Chandra Simhadri**	**1. Refined terminology** **2. Added delimiters**
1.2	**David Fullerton**	**1. Extended protocol** **2. Formatting changes to conform with other networking protocols** **3. General snarkiness**
1.3	**David Matusevich**	**1. Added Version History table** **2. Bug fixes** **3. Changes in sections 2.4 and 2.5** **4. Corrected examples** **5. Added Explanations to Error messages**

Arithmetic Operations Transfer Protocol (AOTP)

1.0. Introduction

1.1. Purpose

AOTP is a protocol for networked systems that enables the computation of arithmetic operations with decimal positive numbers (Base10). The operations supported are Addition, Subtraction, Multiplication and Division. The framework for the operation of this protocol is a client / server architecture.

The scenario of the software architecture above the networking topology is that a software client (called the server) will obtain from the end-user a request for a particular mathematical operation to be performed on a series of numbers. The numbers allowed will be a series with 10 or less members. The four possible arithmetic operations will not be performed by the server, but the server will send to the client the request to perform the arithmetic. This request from the server to the client will utilize the AOTP protocol to be able to properly send the instructions.

After the server receives a properly structured AOTP server-sending communication, the server will perform the arithmetic operation on the numbers. The server will also use the AOTP client-replying communication to provide the resultant single arithmetic numbers back to the client.

A visual representation for the conceptual understanding of the high-level components, behavior, and architecture of the environment for the ATOP protocol (see Figure 1).

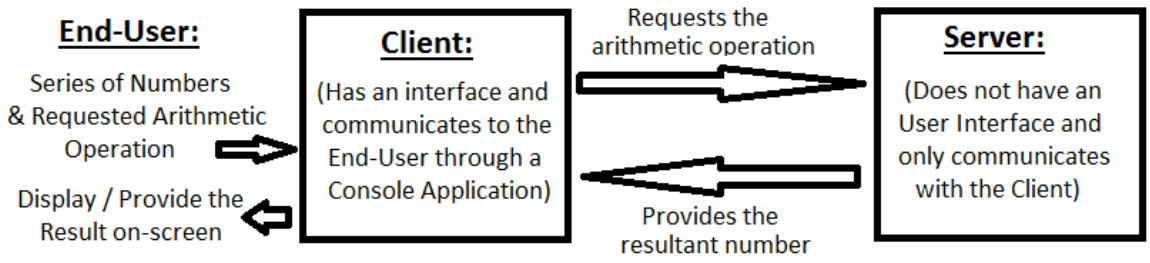

Figure 1. Representation of the End-User Utilizing ATOP in a Client-Server Architecture

1.2. Terminology

The terminology used in this RFC is enumerated below:

- **End-User.** The end-user is the person at the computer who is giving the numbers to the Server software, expecting the server and client to communicate without problems, and provide the arithmetic results in an almost immediate fashion where delays are not detectable.
- **Server.** A computer process that receives Arithmetic operation solving requests from a client, computes the request, and responds to the client.
- **Client.** A computer process that sends a message to sever requesting that it solve an arithmetic operation.
- **Connection.** A transport layer virtual circuit established between the two application programs for the purpose of communication.

1.3. Message Types

The five Message Types defined in this RFC are enumerated below:

1)	**Request Message.**	An AOTP request message is sent the server to the client, requesting that a mathematical operation will be performed. The format is in message format section below (see section 3.1).
2)	**Result Message.**	An AOTP result message is from the client to the server which is providing the resultant of Arithmetic Computed Result (ACR). The message format is provided in a following section (see section 3.2).
3)	**Error Message.**	AOTP supports a list of descriptive error messages that have unique error numbers which are sent from the client to the server which will inform the server about any errors that are found during the computational process for the ACR. Possible errors that can occur are examples like: computational errors, connection oriented errors, etc. These errors are detailed in a following section (see section 3.3).
4)	**CPN Message.**	The Change Port Notice (CPN) is a message that is sent from the server to the client to notify the client about the change in the port number on which the server is listening, instructing it to do a port change. This message format is detailed in a following section (see section 3.4).
5)	**CPN_ACK Message.**	The Change Port Notice Acknowledgement (CPN_ACK) is a message that is sent from the client to the server to notify the server that a change in the port number has been recognized by the client, and the client is now listening on the new port number (see section 3.5)

2.0. Defined Protocol Behavior

There are three types of behavior for the implementation of the client and / or server architecture.

2.1. Normal Operation

The usual operation of the protocol is demonstrated below:

1. The servers starts, listening to a port previously specified by a server administrator
2. A client process sends an AOSR message requesting for computation of an arithmetic operation.
3. The server validates the correctness of the AOSR message by checking the format.

2.2. Error Operation

The operation of the protocol In case of an error is demonstrated below:

1. The server sends an error message* with the error number* to the client and continues listening for new AOSR messages.
2. The client displays the error on the stdout for the user.

2.3. Operation without an Error

If there is no error, then the server computes the requirement and continues with the following process.3.1

2.4. Proper Operation after Changing Ports

The operation of ATOP is as follows when changing the ports:

1. If the server has changed its port:
   The server sends a changing port notice (CPN) to the client and waits for its acknowledgement.
   a. If the client accepts the changing port notice (CPN):
   Client sends an acknowledgement (CPN_ACK) message to sever and client starts receiving on the new port number for the result.
   b. If the client did not acknowledge/accept (CPN_ACK) the CPN:
   The server continues sending the changing port notice (CPN) until an acknowledgement notice (CPN_ACK) is received.

2. If the server cannot change the port
   The server does not send a changing port notice (CPN) to the client.

2.5. Proper Operation a failure during the Changing of the Ports

The proper operation for the system when a failure occurs during the changing of the ports is listed below:

1. If the server did not change the port:
   The server does not send a CPN message. If any Result messages were pending, the server sends the Result message (ACR) and continues listening for new AOSR messages.

2. The client receives the result message (ACR) and displays it on the stdout.

3.0. Definition of Messages.

We define the five categories of messages.

The following five message types will be discussed in the following sections:

3.1) ASOR Message Format

3.2) ACR Message Format

3.3) ERROR* Message Format

3.4) CPN Message Format

3.5) CPN_ACK Message Format

In the following sections we will use SP to signify “space” (ASCII 32) and CRLF to signify “carriage return line feed” (ASCII 13 10).

3.1. ASOR Message Format.

Each message within the ASOR format is comprised of 3 PARTS: operation, operator1, operator2.

ASOR message = ASOR SP operation SP numberbase10 SP numberbase10 CRLF

numberbase10 = d* [.d*]; d = 0|1|2|3|4|5|6|7|8|9|

Example: ASOP<SP>+<SP>2<SP>3<CRLF>

3.2. ACR Message Format.

Each message within the ACR message format is expressed as:

ACR message = ACR SP result CRLF

result = d* [.d*] d = 0|1|2|3|4|5|6|7|8|9|

Example: ACR<SP>5<CRLF>

3.3. ERROR* Message Format.

Each ERROR message within the ATOP protocol is expressed as:

ERROR message = ERROR SP Errorcode CRLF

An example of this format is: <ERRORCODE> which is expressed as:

Errorcode = INVOPERATION|INVOPERATOR| UNRECOGNMESSAGE

INVOPERATION	Operation not recognized. Operation symbol unkown
INVOPERATOR	Operator not recognized. Operator not a number or malformed.
UNRECOGMESSAGE	Message not recognized. Message with no operation, missing operators or without a valid delimiter.

3.4. CPN Message Format.

Each message in the CPN message format is expressed as:

CPN message = CPN SP ddddd CRLF

d = 0|1|2|3|4|5|6|7|8|9|

Example: CPN<SP>50000<CRLF>

3.5. CPN_ACK Message Format

Each message in the CPN_ACK message format is expressed as two possible scenarios:

CPN_ACK message = CPN_ACK SP 1 SP CRLF (1 means to accept the port change)

(or)

CPN_ACK message = CPN_ACK SP 0 SP CRLF (0 means do not accept the port change)

Example: CPN_ACK<SP>1<CRLF>

4.0. Behavior of the Protocol

Various examples are given below to indicate how the ATOP protocol will work in a successful scenario (see section 4.1) along with an unsuccessful scenario (see section 4.2).

4.1. Successful Protocol Communication

The below is an example where the protocol is successful in communicating throughout the duration of the session. This example is a perfect communication that is fully successful without errors.

Step:	CLIENT MESSAGES:	SERVER MESSAGES:
1	AOSR<SP>+<SP>4<SP>12.1<CRLF>	
2		ACR<SP>16.1<CRLF>

4.2. Unsuccessful Protocol Communication

The below is an example where the protocol is NON-successful in communicating throughout the duration of the session.

Step:	CLIENT MESSAGES:	SERVER MESSAGES:
1	*<SP>4<SP>m.1<CRLF>	
2		ERROR<SP>UNRECOGMESSAGE<CRLF>

9.0. Standardized Definitions

The following nomenclature is standardized for all of the definitions within this RFC and shall be used interchangeably, being known as standard and / or equivalent throughout this ATOP protocol.

Def (#)	Terminology:	Standardized Meaning:
1)	**CRLF**	A control line feed, expressed as the ASCII of 13 and 10.
2)	**d**	A single digit, which expressed in regex as: d= 0|1|2|3|4|5|6|7|8|9
3)	**numberbase10**	A number that is represented in $base_{10}$ with one or more digits that might or might not have a a period that is followed by one or more digits, which expressed in regex as: numberbase10 = d* [.d*]
4)	**Operation**	A mathematical operation is one of four symbols for the abbreviation of an arithematic operational instruction. This can be expressed in regex as: Operation = + | - |* | /
5)	**SP**	This token is a single space which is expressed as the ASCII of 32.