
RCSL No: 42-i2504

Edition: March 1984

Author: Carl Henrik Dreyer

Title:

AN IMC-BASED SERVER ACCESS PROTOCOL (SAP)
REPORT

Keywords:

Local Area Networks, Distributed Systems Architecture (DSA),
Inter-Module Communication (IMC), Server Modules, Protocol,
Report, File Transfer, Remote Resource Access.

Abstract:

This paper describes a simple protocol framework, suited for access to a wide range of Service Modules, operating on an IMC-based local area network. Details are given on a file transfer service.

(24 printed pages)

TABLE OF CONTENTS.

PAGE

1.	INTRODUCTION.....	1
2.	BASIC COMMUNICATION SCHEME.....	3
3.	GENERAL BLOCK FORMAT.....	5
4.	DESCRIPTION OF BASIC BLOCKS.....	6
4.1	LOGON blocks.....	6
4.2	REPLY-OK block.....	7
4.3	REPLY-NOTOK block.....	8
4.4	STREAM block.....	8
4.5	STREAM-END block.....	9
5.	FILE TRANSFER REQUESTS.....	10
5.1	CREATE file.....	10
5.2	DELETE file.....	10
5.3	TRANSFER file.....	11
5.4	LOOKUP file (files).....	12
5.5	Format of text files.....	12
5.6	Process communication blocks.....	13
6.	RECOMMANDATIONS.....	15
7.	EXAMPLES OF PROTOCOL SEQUENCES.....	16
7.1	Normal requests and replies.....	16
7.2	Illegal password.....	16
7.3	File transfer.....	17

1. Introduction.

This protocol is designed for use on a local area network, where IMC is used for communication. The Distributed System Architecture (DSA) is supposed, where Modules communicate using IMC.

Some modules offer a service to other modules: file server, printer server etc. Other modules use the service: users. The service offered is in general use of the module resources (e.g. files, or procedures), and in order to gain access to these (or a subset), a user identification (or password) should be given.

The aim of the protocol is to be a common framework to be used for access to any kind of Service module. The frame is deliberately reduced, in order to preserve generality. It contains:

- The USER and SERVER concepts.
- A block format only fixing the first byte as header.
- Defining a few basic blocks.

A further refinement is expected, as systems and needs arise.

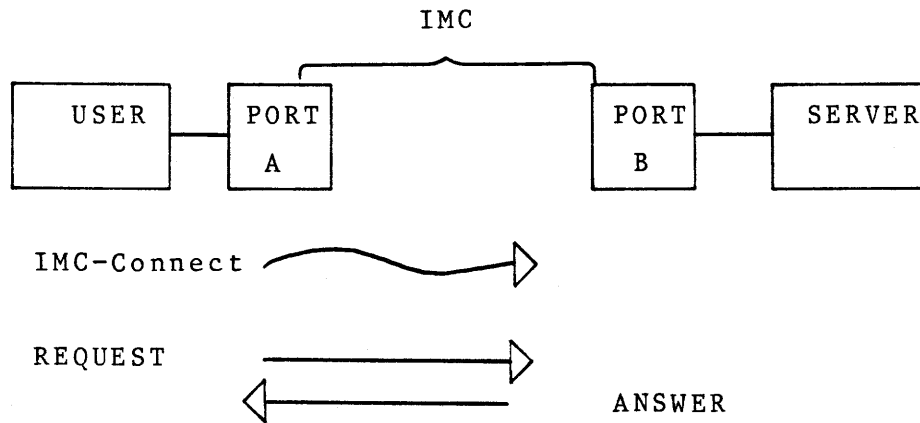
The protocol is based on IMC connections for exchange of blocks. This implies flow control, prevents loss of data blocks, and simplifies addressing.

The protocol is developed for use in file transfer (also supporting RC8000) but is aimed at a more general use.

Characteristics:

- Based on IMC.
- Gaining access from one IMC resident (the USER) to another (the SERVER).
- IMC connections used for access path.
- File transfer.
- Prepared for other services: Remote Ressource Access, Inter Process Communication, Remote Procedure Call, Mailbox Service.

2. Basic Communication Scheme.



Definitions:

USER : The part calling IMC-connect.

SERVER : The part receiving connect by calling IMC-get-connection.

This figure defines the roles of the parties in the dialog. The USER is the active part, using the service given by the SERVER. The USER often acts on behalf of an actual user, or on behalf of a process acting on behalf of a user (etc). The SERVER may use processes for implementing the service, but that is a local matter as well. The USER and SERVER may reside on the same machine, without caring, as IMC is used for addressing.

Note that the roles are defined by which actions are performed to get the connection established. So a module (i.e. a PORT) need not be dedicated for only one role. For example, PORT B above may also be able to connect (calling IMC-connect) to some PORT. On such a connection, it will be USER.

Again, the IMC connection is always established by the USER. The dialogue on the connection is implemented using only IMC-send and -receive of data blocks.

The SERVER is accessed via the PORT, uniquely identified in the network by its name, and the USER just states this name when connecting. Along with the connect operation (sent as first block), the USER sends a LOGON request block, stating a password. The SERVER responds with an REPLY block. These are examples of the basic blocks.

When the LOGON block is replied OK, the dialogue starts: The USER sends a REQUEST, and the SERVER executes it, and sends a REPLY. Only one REQUEST may be issued at a time. This is a simple, "half duplex" scheme, resembling very much a "remote procedure call". The REQUEST specifies some function (procedure) to be executed, and the REPLY states the success: either OK or NOT OK.

This scheme goes for all blocks except STREAM blocks, intended for transfer of bulk data, for example a file. A STREAM has a SENDER end and a RECEIVER end, and these roles must be negotiated in advance, using the REQUEST-REPLY scheme.

When the REQUESTER does not need the service at the SERVER any more, it just disconnects (IMC-call). This eliminates the need for a LOGOFF block.

3. General block format.

!	!	Parameters/data	!
---	---	-----------------	---

↑
└─ HEADER BYTE

The header byte defines the format of the block, and the meaning. It contains a binary number, ranging 0-255. Values 0-15 are reserved for basic blocks, the rest are used for specific services. The possible header byte values (above 15), as well as the format of parameters, are ultimately defined by the SERVER, and the USER must conform with this.

4. Description of basic blocks.

The basic blocks should be used in all implementations of this protocol. The formats should be strictly followed, to ensure compatibility.

Any SERVER should support LOGON, REPLYOK, REPLYNOTOK, but the STREAM blocks are optional.

Note that no LOGOFF block is defined, as the USER just disconnects when it is through.

4.1 LOGON block.

Sent by the USER, as first block on the connection. Max. block length is 128 bytes.

!	0	!	MAX	!	RSV	!	User name/password	!
!		!	BLOCK	!		!		!

Max. block : Two byte integer (INTEL format, LSB first). Max block size used by the USER, in bytes. This is used for a common negotiation of a max. block size.

RSV : One byte. Reserved for future use.

User name/password: ASCII text string. Must conform with the conventions at SERVER. In general, up to 12 characters consisting of digits and small letters should always be allowed. The intention of this field is that the SERVER (via some kind of user catalog) offers the USER specific access rights, sets special parameters, and the like.

4.2 **REPLY-OK block.**

Sent by the SERVER after successful execution of a REQUEST.

!	1	!	Params/data	!
---	---	---	-------------	---

Params/data: Format and content depends on the REQUEST executed, and is specified by the SERVER.

Format of REPLY-OK to LOGON block.

This block format is (like the LOGON block) restricted, to conserve compatibility:

!	1	!	MAX	!	SERV	!		!
							OPTIONAL	
!		!	BLOCK	!	TYPE	!		!

Max. block: Two byte integer. Max. block size used by SERVER, computed as minimum of the size stated in the LOGON block and SERVER's own max. block size.

SERV.TYPE : One byte integer. This number uniquely identifies the service given by the SERVER. Every new service module implemented should be given its own type. A list should be maintained at a central place.

Currently defined types:

0 -31 : File Servers.

 0 : CP/M file server.

 1 : Concurrent CP/M file server.

 2 : XENIX file server.

 3 : RC8000 file server.

32-40 : Teletex applications.

64-255: T.B.D.

4.3 REPLY-NOTOK block.

This block is sent by the SERVER after a non-successfull execution of a REQUEST.

! 2 ! CODE ! STATUS !

CODE : One-byte integer. May be used to give a coded explanation on the problem. If not used, it should be zero.

STATUS : Contains an informative (text string) explanation on the problem, in a form presentable to a (human) user. This field is demanded. Max. length is 32 ASCII characters. A possible human is supposed to know the REQUEST also, so the text may be short. For example, a REQUEST for deleting a file may be REPLY-NOTOK'ed with status "not found", as the rest understood.

4.4 STREAM block.

This block is intended for transfer of bulk data, e.g. a file. It pre-supposes a negotiation on the direction of the stream. It is not acknowledged from the receiving end. If this end disagrees, it disconnects. Max. block size is negotiated in the logon-phase.

! 3 ! Data !

The receiving end must be prepared for receiving STREAM blocks until a STREAM-END block is received. Then the REQUEST-REPLY dialogue is resumed.

4.5 STREAM-END block.

This block terminates a sequence of STREAM blocks. It does not contain any STREAM data itself, but may supply some optional information, e.g. a status text.

! 4 ! Inf. !

5. FILE SERVER REQUESTS.

This chapter describes a set of REQUESTS that should be supported by a File Server. Only a few parameters are specific to an RC8000 file server, just as others may be defined as specific to a XENIX file server.

Though these special parameters are very useful when transferring between identical file systems, a minimum set of functions must be supported, to enable transfer between incompatible file systems.

5.1 CREATE FILE From USER

! 16 ! File name !

Requests creation of the file. File name must conform with the conventions of the SERVER.

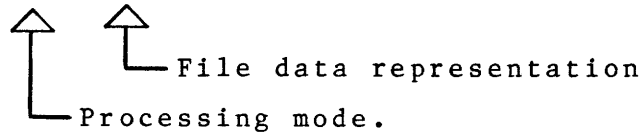
5.2 DELETE FILE From USER

! 17 ! File name !

Requests deletion of the file.

5.3 TRANSFER FILE From USER

! 18 ! ! ! File name !



Requests a file to be transferred. If the reply is OK, the data transfer starts without any further negotiation.

Processing mode (one byte):

R = read file. SERVER is SENDER.
 W = write file. SERVER is RECEIVER.
 A = append file. SERVER is RECEIVER, and file
 data should be appended to the existing file.

File data representation (one byte):

T = Text format. The file data sent must only
 contain characters relevant in the file.
 Section 5.5 contains details.
 B = Binary. Data is in no way restricted, and the
 interpretation is a matter of mutual agreement.
 F = Formatted. Special for RC8000 File Server.
 First 30 bytes of data contains catalog entry
 tail.

File name: Must conform with conventions at the SERVER.

5.4 LOOKUP FILE (FILES)

! 19 ! ! File name mask !



Format of data: N = only names,
B = servers binary format.

Requests a file directory lookup for a file or a group of files, selected by a mask (obeying the conventions at the SERVER).

If reply is OK, a data transfer is initiated, (SERVER is SENDER).

In format N, the data contains the file names found.

In format B, the format is defined by SERVER.

5.5 Format of text files.

As text files is the most obvious format when transferring files between incompatible file systems, some specific details are required.

In general, a file specified as text is just a sequence of bytes. All bytes sent in the STREAM blocks are part of the file, and so no "End of File"-character should be sent, and no "unused disc space" should be sent. The characters are encoded in ASCII code (IA5 alfabeth), but as the transfer is in 8-bit bytes, character values over 127 are accepted (to reduce limitations).

The text file option may be viewed as a facility for automatic conversion between local representation of text. With this approach, the following few conventions are stated (referring to IA5, and ISO 6937 for text communication):

1. No New Line (NL) character is included.

The function is expressed by the sequence CR (value 13) - LF (value 10).

2. The characters EM (value 25) and SUB (value 26) should be avoided in text.

5.6 Process communication blocks.

RC8000 file server may include the following process communication blocks.

OPEN process

!	20	!	!	Process name	!
---	----	---	---	--------------	---



Kind, one byte (binary)

Kind: information on which standard actions should be taken, i.e. process kind (internal, bs, etc.).

The process is selected for communication, but only during the following message requests, i.e. when some other request block is sent, an implicit "close process" is executed.

MESSAGE

! 21 ! Message !

Message is 24 bytes, corresponding to an internal RC8000 message. I/O messages are not supported.

The message is sent to the process previously opened, and the answer is awaited. The REPLY block contains the answer:

! 1 ! Result ! answer !

Where result is one byte, answer is 24 bytes.

6. Recommendations.

Timeouts

The SERVER should have a "long" timeout, so that the connection may be idle for some time, but not infinitely. At timeout a disconnect should be issued.

The USER should have a "short" timeout, when awaiting a reply, and disconnect after timeout.

When awaiting STREAM blocks, the RECEIVER should have a "short" timeout.

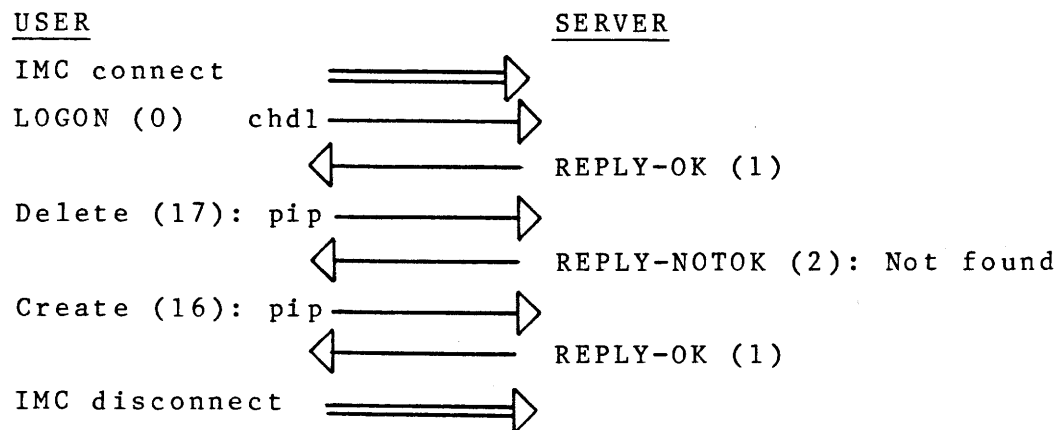
Disconnect

Abnormal disconnect should be done at timeout.

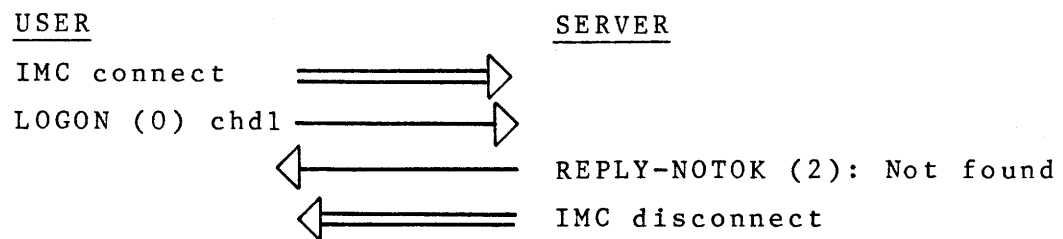
When the SERVER experiences a LOGON with wrong password, it should send REPLY-NOTOK and record the event, and disconnect.

7. Examples of protocol sequences.

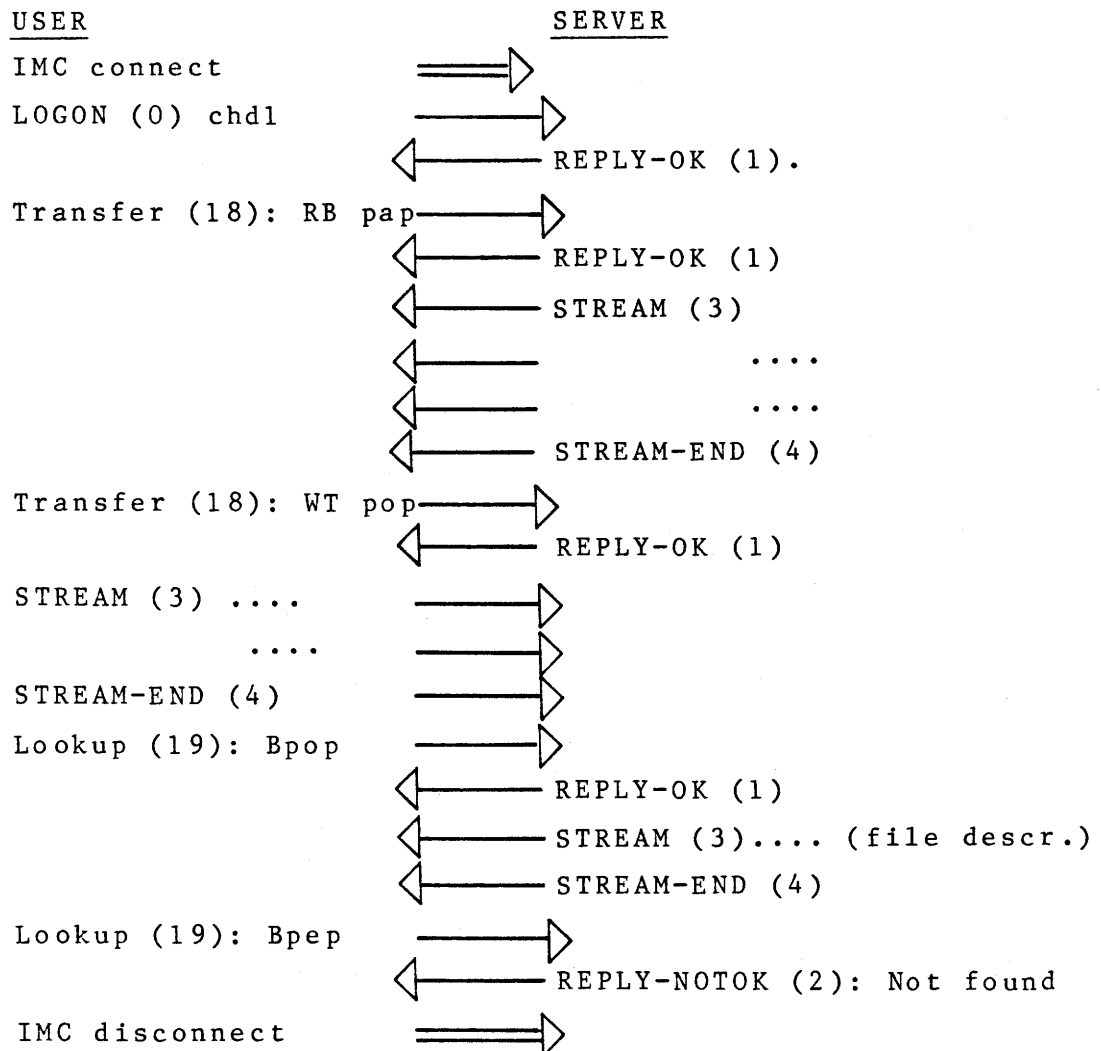
7.1 Normal requests and replies.



7.2 Illegal password



7.3 File transfer



RETURN LETTER

Title: AN IMC-BASED SERVER ACCESS PROTOCOL
(SAP) REPORT.

RCSL No.: 42-i2504

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

Do you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

RCSL Nr. 46-F 0093

..... Fold here

..... Do not tear - Fold here and staple

Affix
postage
here

 **REGNECENTRALEN**
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark