Network Working Group R. Braden
Request for Comments: 740 UCLA-CCN
NIC: 42423 22 November 1977
Obsoletes: 189, 599

NETRJS PROTOCOL

A. Introduction

NETRJS, a private protocol for remote job entry service, was defined and implemented by the UCLA Campus Computing Network (CCN) for batch job submission to an IBM 360 Model 91. CCN's NETRJS server allows a remote user, or a daemon process working in behalf of a user, to access CCN's RJS ("Remote Job Service") subsystem. RJS provides remote job entry service to real remote batch (card reader/line printer) terminals over direct communications lines as well as to the ARPANET.

A batch user at a remote host needs a NETRJS user process to communicate with the NETRJS server at the batch host. An active NETRJS user process simulates a "Virtual Remote Batch Terminal", or "VRBT".

A VRBT may have virtual card readers, printers, and punches. In addition, every VRBT has a virtual remote operator console. Using a virtual card reader, a Network user can transmit a stream of card images comprising one or more batch jobs, complete with job control language ("JCL"), to the batch server host. The NETRJS server will cause these jobs to be spooled into the batch system to be executed according to their priority. NETRJS will automatically return the print and/or punch output images which are created by these jobs to the virtual printer and/or card punch at the VRBT from which the job was submitted. The batch user can wait for his output, or he can signoff and signon again later to receive it.

To initiate a NETRJS session, the user process must execute a standard ICP to a fixed socket at the server. The result is to establish a full-duplex Telnet connection for the virtual remote operator console, allowing the VRBT to signon to RJS. The virtual remote operator console can then be used to issue commands to NETRJS and to receive status, confirmation, and error messages from the

server. The most important remote operator commands are summarized in Appendix D.

Different VRBT's are distinguished by 8-character terminal id's, which are assigned by the server site to individual batch users or user groups.

B. Connections and Protocols

The protocol uses up to five connections between the user and server processes. The operator console uses a a full-duplex Telnet connection. The data transfer streams for the virtual card reader, printer, and punch each use a separate simplex connection under a data transfer protocol defined in Appendix A. This document will use the term "channel" for one of these simplex data transfer connections and will designate a connection "input" or "output" with reference to the server.

A particular data transfer channel needs to be open only while it is in use, and different channels may be used sequentially or simultaneously. CCN's NETRJS server will support simultaneous operation of a virtual card reader, a virtual printer, and a virtual punch (in addition to the operator console) on the same VRBT process. The NETRJS protocol could easily be extended to any number of simultaneously-operating virtual card readers, printers, and punches.

The NETRJS server takes a passive role in opening the data channels: the server only "listens" for an RFC from the user process. NETRJS is defined with an 8-bit byte size on all data channels.

Some implementations of NETRJS user processes are daemons, operating as background processes to submit jobs from a list of user requests; other implementations are interactive processes executed directly under terminal control by remote users. In the latter case, the VRBT process generally multiplexes the user terminal between NETRJS, i.e., acting as the remote operator console, and entering local commands to control the VRBT. Local VRBT commands allow selection of the files containing job streams to be sent to the server as well as files to receive job output from the server. Other local commands would cause the VRBT to open data transfer channels to the NETRJS server and to close these channels to free buffer space or abort transmission.

The user process has a choice of three ICP sockets, to select the character set of the VRBT -- ASCII-68, ASCII-63, or EBCDIC. The server will make the corresponding translation of the data in the card reader and printer channels. (In the CCN implementation of NETRJS, an EBCDIC VRBT will transmit and receive, without

translation, "transparent" streams of 8-bit bytes, since CCN is an EBCDIC installation). The punch stream will always be transparent, outputting "binary decks" of 80-byte records untranslated. The operator console connections always use Network ASCII, as defined by the Telnet protocol.

The NETRJS protocol provides data compression, replacing repeated blanks or other characters by repeat counts. However, when the terminal id is assigned, a particular network VRBT may be specified to use no data compression. In this case, NETRJS will simply truncate trailing blanks and send records in a simple "op code-length-data" form, called "truncated format" (see Appendix A).

C. Starting and Terminating a Session

The remote user establishes a connection to the NETRJS server by executing an ICP to the contact socket 71 (decimal) for EBCDIC, socket 73 (decimal) for ASCII-68, or to socket 75 (decimal) for ASCII-63. A successful ICP results in a pair of connections which are in fact the NETRJS operator console connections. NETRJS will send a READY message over the operator output connection.

The user (process) must now enter a valid NETRJS signon command ("SIGNON terminal-id") through the virtual remote operator console. RJS will normally acknowledge signon with a console message; however, if there is no available NETRJS server port, NETRJS will indicate refusal by closing both operator connections. If the user fails to enter a valid signon within 3 minutes, NETRJS will close the operator connections. If the VRBT attempts to open data transfer channels before the signon command is accepted, the data transfer channels will be refused with an error message to the VRBT operator console.

Suppose that S is the even number sent in the ICP; then the NETRJS connections have sockets at the server with fixed relation to S, as shown in the following table:

Channel	Server Socket	User Socket
Remote Operator Console Input	S	U + 3 Telnet
Remote Operator Console Output	S + 1	U + 2 Telnet
Data Transfer - Card Reader #1	S + 2	any odd number
Data Transfer - Printer #1	S + 3	any even number
Data Transfer - Punch #1	S + 5	any even number

Once the VRBT has issued a valid signon, it can open data transfer channels and initiate input and output operations as explained in the following sections. To terminate the session, the VRBT may close all

connections. Alternatively, it may enter a SIGNOFF command through the virtual remote operator console. Receiving a SIGNOFF, NETRJS will wait until the current job output streams are complete and then itself terminate the session by closing all connections.

D. Input Operations

A job stream for submission to the NETRJS server is a series of logical records, each of which is a card image of at most 80 characters. The user can submit a "stack" of successive jobs through the card reader channel with no end-of-job indication between jobs; NETRJS is able to parse the JCL sufficiently to recognize the beginning of each job.

To submit a batch job or stack of jobs for execution, the user process must first open the card reader channel by issuing an Init for foreign socket S+2 and the appropriate local socket. NETRJS, which is listening on socket S+2, will return an RTS command to open the channel. When the channel is open, the user can begin sending his job stream using the protocol defined in Apendix A. For each job successfully spooled, NETRJS will send a confirming message to the remote operator console.

At the end of the job stack, the user process must send an End-of-Data transaction to initiate processing of the last job. NETRJS will then close the channel (to avoid holding buffer space unnecessarily). At any time during the session, the user process can re-open the card reader channel and transmit another job stack. It can also terminate the session and signon later to get the output.

If the user process leaves the channel open for 5 minutes without sending any bits, the server will abort (close) the channel. The user process can abort the card reader channel at any time by closing the channel; NETRJS will then discard the last partially spooled job. If NETRJS finds an error (e.g., transaction sequence number error or a dropped bit), it will abort the channel by closing the channel prematurely, and also inform the user process that the job was discarded (thus solving the race condition between End-of-Data and aborting). The user process should retransmit only those jobs in the stack that have not been completely spooled.

If the user’s process, NCP, or host, or the Network itself fails during input, RJS will discard the job being transmitted. A message informing the user that this job was discarded will be generated and sent to him the next time he signs on. On the other hand, those jobs whose receipt have been acknowledged on the operator’s console will not be affected by the failure, but will be executed by the server.

E. Output Operations

The VRBT may wait to set up a virtual printer or punch and open its channel until a STATUS message from NETRJS indicates output is ready; or it may leave the output channel(s) open during the entire session, ready to receive output whenever it becomes available. The VRBT can also control which one of several available jobs is to be returned by entering appropriate operator commands.

To be prepared to receive printer (or punch) output from its jobs, the VRBT issues an Init for foreign socket S+3 or S+5 for printer or punch output, respectively. NETRJS is listening on these sockets and should immediately return an STR. However, it is possible that because of a buffer shortage, NETRJS will refuse the connection by returning a CLS; in this case, try again later.

When NETRJS has job output for a particular virtual terminal and a corresponding open output channel, it will send the output as a series of logical records using the protocol in Appendix A. The first record will consist of the job name (8 characters) followed by a comma and then the ID string from the JOB card, if any. In the printer stream, the first column of each record after the first will be an ASA carriage control character (see Appendix C). A virtual printer in NETRJS has 254 columns, exclusive of carriage control; NETRJS will send up to 255 characters of a logical record it finds in a SYSOUT data set. If the user wishes to reject or fold records longer than some smaller record size, he can do so in his VRBT process.

NETRJS will send an End-of-Data transaction and then close an output channel at the end of the output for each complete batch job; the remote site must then send a new RFC to start output for another job. This gives the remote site a chance to allocate a new file for each job without breaking the output within a job.

If the batch user wants to cancel (or backspace or defer) the output of a particular job, he can enter appropriate NETRJS commands on the operator input channel (see Appendix D).

If NETRJS encounters a permanent I/O error in reading the disk data set, it will notify the user via his console, skip forward to the next set of system messages or SYSOUT data set in the same job, and continue. If the user process stops accepting bits for 5 minutes, the server will abort the channel. In any case, the user will receive notification of termination of output data transfer for each job via a remote console message.

If the user detects an error in the stream, he can issue a Backspace (BSP) command from his console to repeat the last "page" of output, or a Restart (RST) command to repeat from the last SYSOUT data set or the beginning of the job, or he can abort the channel by closing his socket. If he aborts the channel, NETRJS will simulate a Backspace command, and when the user re-opens the channel the job will begin transmission again from an earlier point in the same data set. This is true even if the user terminates the current session first and reopens the channnel in a later session; RJS saves the state of every incomplete output stream. However, before re-opening the channel he can defer this job for later output, restart it at the beginning, or cancel its output (see Appendix D). Note that aborting the channel is only effective if NETRJS has not yet sent the End-of-Data transaction.

If the user's process, NCP, or host or the Network itself fails during an output operation, NETRJS will act as if the channel had been aborted and the user signed off. NETRJS will discard the output of a job only after receiving the RFNM from the last data transfer message (containing an End-of-Data). In no case should a NETRJS user lose output from a batch job.

APPENDIX A

Data Transfer Protocol in NETRJS

1. Introduction

The records in the data transfer channels (for virtual card reader, printer, and punch) are generally grouped into transactions preceded by headers. The transaction header includes a sequence number and the length of the transaction. Network byte size must be 8 bits in these data streams.

A transaction is the unit of buffering within the server software, and is limited to 880 8-bit bytes. Transactions can be as short as one record; however, those sites which are concerned with efficiency should send transactions as close as possible to the 880 byte limit.

There is no necessary connection between physical message boundaries and transactions ("logical messages"); the NCP can break a transaction arbitrarily into physical messages. The CCN server starts each transaction at the beginning of a new physical message, but this is not a requirement of the protocol.

Each logical record within a transaction begins with an "op code" byte which contains the channel identification, so its value is unique to each channel but constant within a channel. This choice provides the receiver with a convenient way to verify bit-synchronization, and it also allows an extension in the future to true "multi-leaving" (i.e., multiplexing all channels within one connection in each direction).

The only provisions for transmission error detection in the current NETRJS protocol are (1) the "op code" byte to verify bit synchronization and (2) the transaction sequence number. Under the NETRJS protocol, a data transfer error must abort the entire transmission; there is no provision for restart.

2. Meta-Notation

The following description of the NETRJS data transfer protocol uses a formal notation derived from that proposed in RFC 31 by Bobrow and Sutherland. The notation consists of a series of productions for bit string variables. Each variable name which represents a fixed length field is followed by the length in bits (e.g., SEQNUMB(16)). Numbers enclosed in quotes are decimal, unless qualified by a leading X meaning hex. Since each hex digit is 4 bits, the length is not shown explicitly in hex numbers. For example, '255'(8) and X'FF' both represent a string of 8 one bits.

The meta-syntactic operators are:

| | :alternative string

[ ] :optional string

( ) :grouping

\+ :catenation of bit strings

The numerical value of a bit string (interpreted as an integer) is symbolized by a lower case identifier preceding the string expression and separated by a colon. For example, in "i:FIELD(8)", i symbolizes the numeric value of the 8 bit string FIELD.

Finally, we use Bobrow and Sutherland's symbolism for iteration of a sub-string: (STRING-EXPRESSION = n); denotes n occurrences of STRING-EXPRESSION, implicitly catenated together. Here any n greater or equal to 0 is assumed unless n is explicitly restricted.

3. Protocol Definition

STREAM ::= (TRANSACTION = n) + [END-OF-DATA]

That is, STREAM, the entire sequence of data on a particular open channel, is a sequence of n TRANSACTIONS followed by an END-OF-DATA marker (omitted if the sender aborts the channel).

TRANSACTION ::= THEAD(72) + (RECORD = r) + ('0'(1) = f)

That is, a transaction consists of a 72 bit header, r records, and f filler bits; it may not exceed 880*8 bits.

```
THEAD ::= X'FF'+f:FILLER(8)+SEQNUMB(16)+LENGTH(32)+X'00'
```

Transactions are to be consecutively numbered in the SEQNUMB field, starting with 0 in the first transaction after the channel is (re-) opened. The 32 bit LENGTH field gives the total length in bits of the r RECORD's which follow. For convenience, the using site may add f additional filler bits at the end of the transaction to reach a convenient word boundary on his machine; the value f is transmitted in the FILLER field of THEAD.

```
RECORD ::= COMPRESSED | TRUNCATED
```

RJS will accept intermixed RECORD's which are COMPRESSED or TRUNCATED in an input stream. RJS will send one or the other format in the printer and punch streams to a given VRBT; the choice is determined for each terminal id.

```
COMPRESSED ::= '2'(2) + DEVID(6) + (STRING = p) + '0'(8)

STRING     ::= ('6'(3) + i:DUPCOUNT(5))  |
```

This form represents a string of i consecutive blanks

```
               ('7'(3) + i:DUPCOUNT(5) + TEXTBYTE(8)) |
```

This form represents string of i consecutive duplicates of TEXTBYTE.

```
               ('2'(2) + j:LENGTH(6) + (TEXTBYTE(8) = j))
```

This form represents a string of j characters.

```
TRUNCATED  ::= '3'(2) + DEVID(6) + n:COUNT(8) + (TEXTBYTE(8)=n)

DEVID(6)   ::= DEVNO(3) + t:DEVTYPE(3)
```

DEVID identifies a particular virtual device, i.e., it identifies a channel. DEVTYPE specifies the type of device, as follows:

```
   t = 1:  Output to remote operator console
       2:  Input from remote operator console
       3:  Input from card reader
       4:  Output to printer
       5:  Output to card punch
     6,7:  Unused
```

DEVNO identifies the particular device of type t at this remote site; at present only DEVNO = 0 is possible.

END-OF-DATA ::=X'FE'

Signals end of job (output) or job stack (input).

APPENDIX B

Telnet for VRBT Operator Console

The remote operator console connections use the ASCII Telnet protocol. Specifically:

1. The following one-to-one character mappings are used for the three EBCDIC graphics not in ASCII:

ASCII in Telnet	NETRJS
broken vertical bar	solid vertical bar
tilde	not sign
back slash	cent sign

2. Telnet controls are ignored.

3. An operator console input line which exceeds 133 characters (exclusive of CR LF) is truncated by NETRJS.

4. NETRJS accepts BS (Control-H) to delete a character and CAN (Control-X) to delete the current line. The sequence CR LF terminates each input and output line. HT (Control-I) is translated to a single space. An ETX (Control-C) terminates (aborts) the session. All other ASCII control characters are ignored.

5. NETRJS translates the six ASCII graphics with no equivalent in EBCDIC into the character question mark ("?") on input.

APPENDIX C

Carriage Control

The carriage control characters sent in a printer channel by NETRJS conform to IBM's extended USASI code, defined by the following table:

CODE	ACTION BEFORE WRITING RECORD
Blank	Space one line before printing
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1
2	Skip to channel 2
3	Skip to channel 3
4	Skip to channel 4
5	Skip to channel 5
6	Skip to channel 6
7	Skip to channel 7
8	Skip to channel 8
9	Skip to channel 9
A	Skip to channel 10
B	Skip to channel 11
C	Skip to channel 12

APPENDIX D

Network/RJS Command Summary

This section presents an overview of the RJS Operator Commands, for the complete form and parameter specifications please see references 2 and 3.

Terminal Control and Information Commands

SIGNON	First command of a session; identifies VRBT by giving its terminal id.
SIGNOFF	Last command of a session; RJS waits for any data transfer in progress to complete and then closes all connections.
STATUS	Outputs on the remote operator console a complete list, or a summary, of all jobs in the system for this VRBT, with an indication of their processing status in the batch host.
ALERT	Outputs on the remote operator console an "Alert" message, if any, from the computer operator. The Alert message is also automatically sent when the user does a SIGNON, or whenever the message changes.
MSG	Sends a message to the computer operator or to any other RJS terminal (real or virtual). A message from the computer operator or another RJS terminal will automatically appear on the remote operator console.

Job Control and Routing Commands

Under CCN’s job management system, the default destination for output is the input source. Thus, a job submitted under a given VRBT will be returned to that VRBT (i.e., the same terminal id), unless the user’s JCL overrides the default destination.

RJS places print and punch output destined for a particular remote terminal into either an Active Queue or a Deferred Queue. When the user opens his print or punch output channel, RJS immediately starts sending job output from the Active Queue, and continues until this queue is empty. Job output in the Deferred Queue, on the other hand, must be called for by job name, (via a RESET command from the remote operator) before RJS will send it. The Active/Deferred choice for output from a job is determined by the

deferral status of the VRBT when the job is entered; the deferral status, which is set to the Active option when the user signs on, may be changed by the SET command.

SET — Allows the remote user to change certain properties of his VRBT for the duration of the current session;

(a) May change the default output destination to be another (real or virtual) RJS terminal or the central facility.

(b) May change the deferral status of the VRBT.

DEFER — Moves the print and punch output for a specified job or set of jobs from the Active Queue to the Deferred Queue. If the job's output is in the process of being transmitted over a channel, RJS aborts the channel and saves the current output location before moving the job to the Deferred Queue. A subsequent RESET command will return it to the Active Queue with an implied Backspace (BSP).

RESET — Moves specified job(s) from Deferred to Active Queue so they may be sent to user. A specific list of job names or all jobs can be moved with one RESET command.

ROUTE — Re-routes output of specified jobs (or all jobs) waiting in the Active and Deferred Queues for the VRBT. The new destination may be any other RJS terminal or the central facility.

ABORT — Cancels a job which was successfully submitted and awaiting execution or is currently executing.

Output Stream Control Commands

BSP (BACKSPACE) — "Backspaces" output stream within current sysout data set. Actual amount backspaced depends upon sysout blocking but is roughly equivalent to a page on the line printer.

CAN (CANCEL) — (a) On an output channel, CAN causes the rest of the output in the sysout data set currently being transmitted to be omitted. Alternatively, may omit the rest of the sysout data sets for the job currently being transmitted; however, the remaining

system and accounting messages will be sent.

(b) On an input channel, CAN causes RJS to ignore the job currently being read. However, the channel is not aborted as a result, and RJS will continue reading in jobs on the channel.

(c) CAN can delete all sysout data sets for specified job(s) waiting in Active or Deferred Queue.

RST (RESTART) (a) Restarts a specified output stream at the beginning of the current sysout data set or, optionally, at the beginning of the job.

(b) Marks as restarted specified job(s) whose transmission was earlier interrupted by system failure or user action (e.g., DEFER command or aborting the channel). When RJS transmits these jobs again it will start at the beginning of the partially transmitted sysout data set or, optionally, at the beginning of the job. This function may be applied to jobs in either the Active or the Deferred Queue; however, if the job was in the Deferred Queue then RST also moves it to the Active Queue. If the job was never transmitted, RST has no effect other than this queue movement.

REPEAT Sends additional copies of the output of specified jobs.

EAM Echoes the card reader stream back in the printer and/or punch stream.

APPENDIX E

NETRJS TERMINAL OPTIONS

When a new NETRJS virtual terminal is defined, certain options are available; these options are listed below.

1. Truncated/Compressed Data Format

   A VRBT may use either the truncated data format (default) or the compressed format for printer and punch output. See Reference 9 for discussion of the virtues of compression.

2. Automatic Coldstart Job Resubmission

   If "R" (Restart) is specified in the accounting field on the JOB card and if this option is chosen, RJS will automatically resubmit the job from the beginning if the server operating system should be "coldstarted" before all output from the job is returned. Otherwise, the job will be lost and must be resubmitted from the remote terminal in case of a coldstart.

3. Automatic Output RESTART

   With this option, transmission of printer output which is interrupted by a broken connection always starts over at the beginning. Without this option, the output is backspaced approximately one page when restarted, unless the user forces the output to start over from the beginning with a RESTART command when the printer channel is re-opened and before printing begins.

4. Password Protection

   This option allows a password to be supplied when a terminal is signed on, preventing unauthorized use of the terminal ID.

5. Suppression of Punch Separator and Large Letters.

   This option suppresses both separator cards which RJS normally puts in front of each punched output deck, and separator pages on printed output containing the job name in large block letters. These separators are an operational aid when the ouptut is directed to a real printer or punch, but generally undesirable for an ARPA user who is saving the output in a file for on-line examination.

APPENDIX F

Character Translation by CCN Server

A VRBT declares its character set for job input and output by the initial connection socket it chooses. A VRBT can have the ASCII-68, the ASCII-63, or the EBCDIC character set. The ASCII-63 character mapping was added to NETRJS at the request of users whose terminals are equipped with keyboards like those found on the model 33 Teletype.

Since CCN operates an EBCDIC machine, its NETRJS server translates ASCII input to EBCDIC and translates printer output back to ASCII. The details of this translation are described in the following.

For ASCII-68, the following rules are used:

1. There is one-to-one mapping between the three ASCII characters broken vertical bar, tilde, and back slash, which are not in EBCDIC, and the three EBCDIC characters vertical bar, not sign, and cent sign (respectively), which are not in ASCII.

2. The other six ASCII graphics not in EBCDIC are translated on input to unused EBCDIC codes, shown in the table below.

3. The ASCII control DC4 is mapped to and from the EBCDIC control TM.

4. The other EBCDIC characters not in ASCII are mapped in the printer stream into the ASCII question mark.

For ASCII-63, the same rules are used except that the ASCII-63 codes X’60’ and X’7B’ - X’7E’ are mapped as in the following table.

EBCDIC		ASCII-68 VRBT		ASCII-63 VRBT	
vertical bar	X’4F’	vertical bar	X’7C’	open bracket	X’5B’
not sign	X’5F’	tilde	X’7E’	close bracket	X’5D’
cent sign	X’4A’	back slash	X’5C’	back slash	X’5C’
underscore	X’6D’	underscore	X’5F’	left arrow	X’5F’
.	X’71’	up arrow	X’5E’	up arrow	X’5E’
open bracket	X’AD’	open bracket	X’5B’	.	X’7C’
close bracket	X’BD’	close bracket	X’5D’	.	X’7E’
.	X’8B’	open brace	X’7B’	.	X’7B’
.	X’9B’	close brace	X’7D’	.	X’7D’
.	X’79’	accent	X’60’	.	X’60’

APPENDIX G

REFERENCES

1. "Interim NETRJS Specifications", R. T. Braden. RFC #189: NIC #7133, July 15, 1971.

This was the basic system programmer's definition document. The proposed changes mentioned on the first page of RFC #189 were never implemented, since the DTP then in vogue became obsolete.

2. "NETRJS Remote Operator Commands", R. T. Braden. NIC #7182, August 9, 1971

This document together with References 3 and 8 define the remote operator (i.e. user) command language for NETRJS, and form the basic user documentation for NETRJS at CCN.

3. "Implementation of a Remote Job Service", V. Martin and T. W. Springer. NIC #7183, July, 1971.

4. "Remote Job Entry to CCN via UCLA Sigma 7; A scenario", UCLA/CCN. NIC #7748, November 15, 1971.

This document described the first NETRJS user implementation available on a server host. This program is no longer of general interest.

5. "Using Network Remote Job Entry", E. F. Harslem. RFC #307: NIC #9258, February 24, 1972.

This document is out of date, but describes generally the Tenex NETRJS user process "RJS".

6. "EBCDIC/ASCII Mapping for Network RJS", R. T. Braden. RFC #338: NIC #9931, May 17, 1972.

The ASCII-63 mapping described here is no longer correct, but CCN's standard ASCII-68/EBCDIC mapping is described correctly. This information is accurately described in Appendix F of the current document.

7. "NETRJT--Remote Job Service Protocol for TIP's", R. T. Braden. RFC #283: NIC 38165, December 20, 1971.

This was an attempt to define an rje protocol to handle TIPs. Although NETRJT was never implemented, many of its features are incorporated in the current Network standard RJE protocol.

8. "CCN NETRJS Server Messages to Remote User", R. T. Braden. NIC #20268, November 26, 1973.

9. "FTP Data Compression", R. T. Braden. RFC #468: NIC #14742, March 8, 1973.

10. "Update on NETRJS", R. T. Braden. RFC #599: NIC #20854, December 13, 1973.

This updated reference 1, the current document combines the two.

11. "Network Remote Job Entry -- NETRJS", G. Hicks. RFC #325: NIC 9632, April 6, 1972.

12. "CCNRJS: Remote Job Entry between Tenex and UCLA-CCN", D. Crocker. NUTS Note 22, [ISI]<DOCUMENTATION>CCNRJS.DOC, March 5, 1975.

13. "Remote Job Service at UCSB", M. Krilanovich. RFC #477: NIC #14992, May 23, 1973.