Bolt Beranek and Newman Inc.

Report No. 3782

Message Technology Research and Development

Quarterly Progress Report No. 7, 2 July 1977 to 2 October 1977

March 1978

Prepared for:
Defense Advanced Research Projects Agency

(14) BBN-3782

(9) Quarterly progress rept. no. 3; 2 Jul - 2 Oct 77;

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN REPORT NO. 3782	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MESSAGE TECHNOLOGY RESEARCH AND DEVELOPMENT.	5. TYPE OF REPORT & PERIOD COVERED: 7/2/77 - 10/2/77
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): J. Burchfiel T. Myer	8. CONTRACT OR GRANT NUMBER(s): MDA903-76-C-0212, ARPA Order-3161
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (12) 36p
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: March 1978
	13. NUMBER OF PAGES: 32 + ii
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3161.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Hermes
Message Processing
Tenex Security
CINCPAC Test

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes BBN efforts in the continuing development of the HERMES message-processing system, with respect to system design, security requirements and preparations for the DARPA/NAVY/CINCPAC interactive test.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060 100

BBN REPORT NO. 3782

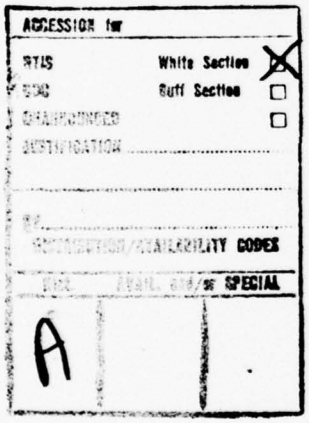
March 1978

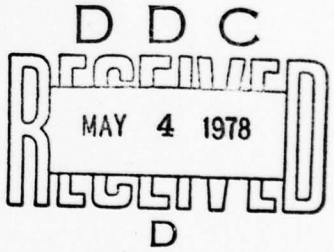
12

MESSAGE TECHNOLOGY RESEARCH AND DEVELOPMENT

Quarterly Progress Report No. 7

2 July 1977 to 2 October 1977





This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3161 Contract No. MDA903-76-C-0212

TABLE OF CONTENTS

1. INTRODUCTION

This report covers progress in message technology under the contract "Message Technology Research and Development", Contract No. MDA903-76-C-0212 for the period 2 July through 2 October 1977.

Hermes System Support

During the July through September quarter, we implemented and released successive versions of Hermes, incorporating new features and demonstrating improvements in speed.

We implemented and released versions of HERMES 4.0.17 and 4.0.22 on the BBN and SRI host computers. These versions were also installed for general use on the four ISI host computers, ISI, ISIB, ISIC and ISIE. Hermes now incorporates efficiency improvements accomplished during the previous quarter. Preliminary analysis suggests a nearly threefold improvement in responsiveness over the January 1977 version.

Documentation

We continued revision and updating of the on-line documentation for the HERMES 4.0 system. All the topics for the DESCRIBE command were brought up to date for HERMES 4.0.22.

Data Management

We continued study and experimentation of the use of HERMES as a tool for office automation and data management.

H1 System

We developed the H1 system, a simplified version of HERMES, consisting of a subset of the most frequently used HERMES commands for message reading and composition, which is upward compatible with full scale HERMES.

Design for a Secure Message System

We completed the preliminary design for a "second-generation" message system with military security features and distributed architecture.

The Architecture of Message Systems

We continued work on a distributed architecture for message systems by further development of the experimental front end module with focus on the command scanner portion. The command scanner employs augmented transition network (ATN) techniques to define the syntax and semantics of a command language

The Message Technology Laboratory

We continued work on the message technology laboratory. We established an enriched set of primitives in the supporting

software, and then created a first-stage programming language that makes it possible to put the new primitives to use in creating extensions to the system. We tested the language with a number of trial applications including templates with conditional formatting behavior dependent on the content of the message to be formatted.

The programming language was also intended to support various types of automated message processing. We tested this assumption by creating an experimental questionnaire application in which a questionnaire-message solicits information and then packages the result in a return message to the originator.

2. HERMES SYSTEM SUPPORT

2.1 HERMES 4.0.17

On June 25, 1977, HERMES 4.0 17 was implemented and released for general use by DARCOM and other users through the ARPANET on the BBN host computers, BBNA, BBNB, BBN, BBND and BBNE, and on the SRI-KA and SRI-KL host computers under the control of access lists. Following its successful performance during July and August, HERMES 4.0.17 was implemented and released for general use on the ISI host computers ISI, ISIB, ISIC and ISID on August 30, 1977. HERMES 4.0.22 was released on September 10, 1977, at all sites under the name of NEWHERMES.

Throughout this quarter, careful revision of the HERMES code resulted in changes that produced major improvements in speed over that of HERMES 4.0.12. These changes were incorporated in successive versions of HERMES, in an orderly progression from the developmental version, which is available only to the HERMES group at BBN, on an experimental basis, through NEWHERMES, which is used by a selected group of regular users. to the regular HERMES.

New features incorporated in HERMES 4.0.17 reflected the improvements developed for the MME HERMES during the first half of 1977. These were described in the previous progress report, BBN Report No. 3618.

2.2 NEWHERMES 4.0.22

HERMES 4.0.22 was released as NEWHERMES at all sites on September 11, 1977. New features in 4.0.22 represented refinements of 4.0.17 from the point of view of user functionality rather than fundamental changes.

2.2.1 The SAVEPARSE Switch

When you QUIT or EXIT from HERMES, or when you GET a new message-file, should HERMES preserve the parse of the message-file you are leaving?

The new SAVEPARSE switch allows you to discard the "parse" portions of the "parseq" (up-arrow) files associated with your message-files, without losing any named sequences you have created.

The "parseq" (up-arrow) file is usually reduced to 3 pages (possibly 1 or 2 pages more if there are voluminous sequences).

Since HERMES now automatically discards bad parses and, since the cost of regenerating the parse is high, in CPU charges and in real time, most users will want to leave the SAVEPARSE switch set to "Yes." As originally implemented in 4.0.22, the SAVEPARSE switch had only two settings, "Ask" and "Yes."

However, some users who have extremely limited disk space, or who never save large message-files, or who seldom access the same message-file more than once, requested the "No" setting, and, therefore, this setting was incorporated in HERMES 4.0.22.

If the SAVEPARSE switch is set to No, HERMES warns you when a parse is discarded.

2.2.2 The PRINT-FORM Template

The new PRINT-FORM template installed in HERMES 4.0.17 proved to be deficient because it made no provision for user-fields. Accordingly, PRINT-FORM was changed in HERMES 4.0.22.

```
PRINT-FORM
  (1) Message-no.:+ Char-Count: Rcvd-Date: "  " Status:+
  (2) Other
  (3) From:+
  (4) To:+
```

```
(5) Cc:+
(6) Bcc:+
(7) Subject:+
(8) In-Reply-To:+
(9) User-Fields:+
(10)
(11) Text:+
```

The template item "User-Fields" prints all user-created fields just as they appear in the original message.

2.2.3 Names of PARSEQ Files

The names of the auxiliary "parseq" files were changed from the form {MESSAGE.TXT};1 to ^MESSAGE.TXT^. The reason for the change is that the curly brace characters, {}, are a part of the set of lower case characters, and hence cannot be typed on an all upper case terminal.

HERMES now automatically converts from the old to the new form of parseq names.

2.2.4 Documentation

The on-line documentation material accessed by the OUTLINE DESCRIBE EXAMPLE and DOCUMENT commands was revised and brought up to date for HERMES 4.0.22.

3. THE H1 SYSTEM

In September, we tested the first experimental implementation of the H1 System, which is a "Junior Version" of HERMES. This system consists of 16 functional commands and 5 informational commands. All commands operate at top command level. These commands which are listed in Appendix A, form a proper subset of HERMES commands.

There are no second-level editors. The fields in the draft message cannot be edited once the user has typed the terminating carriage return or Control-E.

The SEND and ERASE logic has been altered slightly to conform to the lack of a message editor.

Although the new simplified system totally lacks facilities for creating and editing objects such as templates, filters and user-fields, it is completely compatible with full-size Hermes.

We contemplate a mode of use in which user profiles will be set up by HERMES application specialists who use full-size HERMES.

Users who choose to read and create messages using the H1 will have the advantage of a HERMES environment of switch settings, and templates, etc, tailored to their needs. However, their set of commands will be limited to commands required for message processing. We believe that the H1 can be expected to have three advantages for the ordinary user.

1. It will be easier to learn and use because the set of commands is small and largely self-explanatory, e.g., PRINT, FILE, SHOW.

2. The program will be somewhat smaller in overall size.

3. The speed should be greater, because of the smaller program size. Start-up time appears to be faster in this preliminary version.

The lack of a message-editor in this version is frankly experimental. It may prove to be too limiting to be accepted by the user community. We plan to continue to experiment to find the levels of acceptable functionality that can be implemented as HERMES subsets.

4. DESIGN STUDY FOR A SECURE MESSAGE SYSTEM

On September 13 through 16, 1977, Theodore H. Myer and James R. Miller of BBN attended a meeting held at the National Bureau of Standards, Gaithersberg, Md., to plan the scope and content of the study effort of the next generation of message systems.

BBN presented a proposed design for a new message system to meet the requirements of military message traffic, following guidelines established by ARPA.

1. To be implemented after FY 1978.
2. To be highly modular and reliable.
3. To meet requirements for military security.
4. To aim to include facilities for a data base management system.

Significant progress was made during the meeting. BBN and other participants came to substantial agreement on the following points:

1. The proposed message system should be based upon mini computers such as the PDP-11.
2. The system should be based on a secure data kernel such as the secure UNIX system now under development.
3. There should be little or no expansion of functionality beyond the current message systems.
4. Terminals and processors that are geographically close to each other should be interconnected via local very high speed (3 mega baud) networks.
5. The prototype should be available in 3 to 4 years.

4.1 THE SECURITY KERNEL

The heart of the proposed message system is a "Data Security Kernel," which is expected to be available from a source other than BBN, and which will be independently certified as capable of rigorously enforcing security regulations. The Security Kernel will enforce access restrictions for the objects and operations of the next layer of the total system, the File System.

4.2 THE FILE SYSTEM

The File System, running under the Security Kernel, provides the access to the multi-level secure objects, including messages, under access restrictions enforced by the Security Kernel. In addition, the File System supplies discretionary access controls, based upon combinations of capabilities and access control lists.

Capabilities are the rules that govern the operation of the File System and the objects that it contains. Examples of capabilities are the capability to create, delete or copy objects, the capability to change the ACL and the capability to create new capabilities.

Access to a capability is granted by an Access Control List (ACL), which lists the names of the message system users and the rights assigned to each user.

The File System is the source of the primitive objects and operations that are available to the Operating System.

4.3 THE OPERATING SYSTEM

The fundamental underlying structure of the message system is supplied by the Operating System, which uses the File System primitives to build tools for handling the message system objects.

4.4 THE MESSAGE SYSTEM

The Message System takes the tools provided by the Operating System and combines them to form Message System functions.

4.5 USER LANGUAGES

A series of User Languages will be devised to act as "front ends" to interface between the user and the Message System functions. These languages will be easily modifiable, and it is expected that users will be able to select from among several alternative languages to express their requests to the Message System.

4.6 FILE SYSTEM DESIGN

We provided a design for a File System, based upon multi-level secure objects and operations upon these objects.

The design would be subject to the control of the Security Kernel, and in addition, would operate under ACL restrictions. Operations would be defined by Capabilities assigned to objects and would operate under the simple security rule (SSR), which

states that a user at a given security level cannot see anything above the current level. In addition, certain operations would have the property which states that the user is not allowed to write at a security level lower than the current level.

5. THE ARCHITECTURE OF MESSAGE SYSTEMS

During this reporting period, we also continued experimental work on distributed message systems. We suggested that a system could be split into several types of modules.

1. The Front End, contains modules for (a) the terminal and (b) the display handler. This determines the commands and other system functions that are presented to the user.

2. The Intermediate Modules, which may also be considered as part of the Front End, process input and may include (a) I/O handler, (b) command scanner, (c) grammar, and (d) command expeditor.

3. The Back End Processes do the actual work of the system.

5.1 EXPERIMENTAL FRONT END MODULE

We continued implementation of an experimental Front End module. Concentrating on the development and implementation of the command scanner, because of its immediate utility. In order to test the command scanner, it has been necessary to build a rudimentary display handler which acts like a hard-copy terminal.

The command scanner is also dependent on the existence of a grammar for it to follow. In order to facilitate ease in building grammars, a compiler for the command scanner has been developed.

The command scanner employs augmented transition network (ATN) techniques to define the syntax and semantics of a command language. During this quarter we completed::

1. A software module that compiles external descriptions of a command language into a suitable internal ATN form.

2. Support the scanning software that interprets user inputs in the light of a particular ATN.

3. Linkage software for coupling to a back end processing module. We tested the front end module with several hypothetical input languages, including a subset of the hermes language.

5.2 COMMAND LANGUAGE

The command language is viewed as a series of states, each associated with one or more arcs which may have conditions and/or actions associated with them, and is associated with a unique succeeding state. To store parse information, and to supply the scanner with knowledge of what has already occurred, there is a set of "registers" which are operated upon by the actions of the command scanner. Each register is, in fact, the top node of a tree containing items placed into that register by the scanner.

5.3 COMMAND SCANNER

The command scanner recognizes several types of input. A current limitation is that only one type may be associated with one arc. The input types are:

a) Table entry· The scanner will attempt to match input against a table of character strings. Incomplete matches are allowed only if they are nonambiguous, that is, only if one and only one incomplete match is found. On request, the scanner returns extension of the incomplete string up to the first ambiguity or a list of possible matches against the incomplete input. If there is no possible match, the scanner indicates that, and also returns the point at which the input failed to match at all.

b) Number: The scanner expects to see numeric input in a predefined range. While it cannot complete the type-in, it can return a description of the range the number must fall into. It returns an error indication as soon as a mistake is found, e.g., a decimal point in an integer, a minus sign when only positive numbers are acceptable, or a number out of range.

c) Quoted string: The scanner will demand that both the first and last characters be quotes, and that the last character be not followed by a quote. As with numbers, the scanner cannot complete the entry, but on request it returns a description of a quoted string.

d) Routine: The scanner supplies the input string and starting position in that string to a user written routine whose responsibility is to handle all checking and help and error returns.

e) Any: This is similar to quoted string. The scanner will accept any string terminated by a predefined terminating character. The scanner will supply a description of what it expects as a terminating character on request.

f) Default: There may be no more than one default arc associated with one state. This arc is taken at user request. It may or may not generate a string to be added to the input buffer, and may or may not duplicate the actions of another arc in the same state.

g) In-case-of-failure: Planned, but not yet implemented, the in-case-of-failure arc would be taken if all others fail. It might include a spelling correcter.

h) Push: Push is really a subroutine call to a set of states. As such, it has no real characteristics of its own. When the subroutine call is finished, the grammar "pops" back to the calling state and takes the arc associated with the push.

Associated with each arc is a set of terminating characters. There may be up to 36 of these defined for the grammar, and one or all may be acceptable for any state. These characters may fall into two classes for any arc: regular terminators and special terminators. At first, we believed that the special characters should be used to indicated that the next state should

be identical with the current one; upon discovering some counter-examples, the special characters were redefined to be considered by themselves as input for the next state. Thus the input

A&B

can be equivalent to

A and B

simply by defining "&" (the ampersand) to be a special terminator, and "space" to be a regular terminator, and having "&" and "and" appear as table entries which lead to the same next state. In order to permit flexibility, the ampersand arc may not require a terminating character after the ampersand (even though it may have one defined).

Associated with each arc may be a set of actions to perform. One or more registers may be set or cleared, one or more user supplied subroutine calls may be made. Registers may hold information of use to either the back end processes or to the command scanner itself.

Also associated with each arc may be a set of conditions. Thus, even if the user were to type in "correct" input, if the conditions are not met that arc cannot be taken. This decision may be based on the contents of the registers (in the current version, it always is) and is made before doing any evaluation of input. The combination of actions and conditions is extremely powerful, as it permits order free input of arguments, if wished

by the writer of the grammar. It also allows for compactness in the representation of the grammar.

Another feature (as yet not fully implemented) is that of precedence. By associating a precedence level with each arc, ambiguous situations may be resolved by taking the arc whose precedence is greater. Precedence may also be used to control easily the size of the command set available to the user. When the precedence threshold is raised, lower precedence arcs cease to become available, and the user has a more basic set of commands with which to work. The threshold might be under the control of the user or under the control of the system builder, depending on the circumstances.

6. MESSAGE TECHNOLOGY LABORATORY

We began development of a "laboratory" environment to support rapid development and subsequent iteration of message systems and ideas (abstractions appropriate for message related problems as well as formal specification of useful objects and operations). We have already built one experimental system in this laboratory based on the ideas of programmability and the ability to send programs in messages. Initial experience with this experiment has convinced us of the message Technology Laboratory's value as a source of new military message technology.

6.1 ORIENTATION

It is the purpose of the Message Technology Laboratory to serve as a base for the design and implementation of experimental message processing applications.

Currently, the Message Technology Laboratory has been used to construct an experimental message system, R2D2, which goes well beyond HERMES. It is based on the current TENEX paradigm for mail delivery. It also is based on a message oriented programming language, through which applications implementors can specify specialized functions and automated procedures, and is based on the ability to send programs in messages and have these instructions "activated" (evaluated) on receipt of the message. A receiver can arrange to have the programs contained in incoming

messages automatically extracted and executed. Thus a sender can cause complex activity to take place in a receiver's environment.

R2D2 extends Hermes in other ways as well. We have built a storage mechanism for objects other than messages which exist in R2D2. These objects can be closely associated with a message file, creating an environment for the messages in that file. This environment consists of:

1. The message file itself (i.e. the messages in the file)
2. Subsets of that set of messages (domains)
3. Messages in preparation (drafts)
4. Instructions for selecting messages from domains (selectors)
5. Instructions for how to print messages (templates)
6. Instructions for how to compose messages (ctemplates)
7. Composite structures comprising commands and control structure (commands)

User-created objects are named and associated with a message file. This allows certain templates or ctemplates, for example, to be in the user's active environment only if a certain message file is "loaded in". R2D2 maintains a special message file for each user which acts as that user's profile. Several mail files (and associated objects) can be active in the environment at any given time.

In R2D2, printing and composing templates are regarded as sets of instructions (programs) whose execution will result in a message being printed or composed. These instructions form a language

which includes conditionals, testing for field existence or contents, indirection (like calling a subroutine -- having one template invoke another from the environment), and the ability to "call" an object of the other form (a print template can invoke a compose template and vice versa). In addition, the language allows templates to be extracted from the message being printed. It also provides a mechanism for columnar printing.

That part of the language which deals with composition is even richer. Included are replacement, deletion, or appending to fields of the message being created. Text to be used can come from the user, a literal string, or other messages. Also, since some acts of composing (for example, forwarding or replying) use an initial set of existing messages in the creation of new messages, various control structures are included in the language to permit iteration over those existing messages.

6.2 APPLICATIONS

Within R2D2 we have implemented a number of applications arising from current and proposed behaviors of Hermes.

1. We have simulated the actions of the Reply and Forward commands of Hermes. These functions in Hermes carry out a fixed, sophisticated activity. Hermes users have requested the ability to tailor this behavior to suit their own individual needs. The implementation of these functions in R2D2 is achieved by expressing each of them as a program in the relevant subset of MAIL. By modifying these programs, it is now possible for users to tailor the system to their own needs.

2. We have implemented the ability for originators of messages to specify the format in which a message is to be printed on the recipient's terminal. This behavior is

achieved by placing printing templates into the message itself, and arranging with the receiver that his normal printing template will extract and use any such instructions appearing in incoming messages.

3. We have implemented a rudimentary questionnaire facility. A special message is prepared and transmitted to a respondent. The respondent's message system in the same was as that of the previous example, interpreting the instructions in the message. In this case, the instructions prompt the user through a series of questions, and compose a draft message containing the respondent's answers to those questions. This draft is then transmitted back to the originator of the questionnaire.

A demonstration of the user's view of an Message Technology Laboratory scenario embodying some of these functions is shown in Appendix B.

To support R2D2 and the applications which have been implemented in it, the Message Technology Laboratory provides preliminary versions of two capabilities: an environment to support the creation of applications, and an environment for invoking those applications. The former environment supports the applications implementor in his application specification task. The latter is what is usually referred to as a user interface for the applications.

The application creation environment in R2D2 is constructed using many of the capabilities provided by INTERLISP. In particular, all of the facilities provided by the INTERLISP Programmer's Assistant are available.

To being the exploration of user interfaces, a grammar-based command language processor was created and interfaced to R2D2.

The particular grammar created for use with R2D2 was Hermes-like, having extension, recognition, prompting, defaults, and so on. However, a wide variety of front-ends could be built using the same grammar-based parser.

Thus, the current version of the Message Technology Laboratory contains preliminary versions of all of the aspects that we envision being in it: a message processing language, a supporting message application building environment, a message system application, and a user interface.

Appendix A

H1 COMMANDS

FUNCTIONAL COMMANDS

Compose
Delete
  D = Delete
File
Forward
Get
List
Print
Logout
Quit
Reply
Survey
  S = Survey
Undelete
<linefeed>
^

INFORMATIONAL COMMANDS

Describe
Example
Help
Show
Status

There is no message editor. All commands are at the same level. Compose, Reply and Forward prompt the user to create a message, then ask Send? If the user answers Yes, the message is sent and the draft message is erased, if No, the draft is simply erased, and cannot be recovered.

The SHOW command displays the contents of sequences, but the names only of the HERMES filters and templates. The settings of switches, FDESTINATION and LDESTINATION are not shown. These, as well as contents of the current objects, and the existence of user-created objects are controlled by the user's profile. This profile can be changed only through the use of full-size HERMES.

Appendix B

A MESSAGE TECHNOLOGY LABORATORY DEMONSTRATION

This is an excerpt from a demonstration of the experimental message system "R2D2" which was implemented in INTERLISP and demonstrated at Bolt Beranek and Newman.

The message-file <HERMES>CDEMO.TXT.1 used in the demonstration contains seven messages in the DOMAIN named CDEMO.

1 A SURVEY COMMAND CONTROLLED BY MESSAGE CONTENT

These messages are first SURVEYED, using the STEMPLATE. The STEMPLATE prints all messages through the normal survey template, except that Message No. 2 is printed through a FLASH TEMPLATE.

Next, the "FLASH" message is PRINTED using the PTEMPLATE, and we see that it contains a priority field with the contents "FLASH".

Finally, the three templates STEMPLATE, SURVEY, and FLASH are displayed.

Demonstration No. 1

See next page.

```
< SURVEY * CDEMO
    1 CDEMO  6-Sep-77 The RFC
                      Some comments by three people.

************  F L A S H  **************

    2 CDEMO  6-Sep-77 Welcome to R2D2

****************************************
    3 CDEMO  6-Sep-77 This is a surprise about our national pastime.
    4 CDEMO  4-Jan-78 The Atlantic Conflict
    5 CDEMO  4-Jan-78 The Atlantic Conflict
    6 CDEMO  4-Jan-78 The Atlantic Conflict
    7 CDEMO  4-Jan-78 CHOPping is completed.
                      The Atlantic Conflict

< PRINT 2 CDEMO
Date:   6 Sep 1977 1351-EDT
Priority:  FLASH
From:  VITTAL
Subject:  Welcome to R2D2

Friends:

This is a file of several messages which are designed to show you
some of the features of R2D2, a laboratory for experimenting with
new message processing facilities.

-------


< SHOW STEMPLATE
Template:
   STEMPLATE (associated with) PROFILE
      if field PRIORITY contains "FLASH"
        then use template FLASHTEMPLATE
        else use template SURVEY-TEMPLATE

< SHOW FLASHTEMPLATE
Template:
   FLASHTEMPLATE (associated with) PROFILE
      NEWLINE
      "**************  F L A S H  **************" NEWLINE
      NEWLINE
      use template SURVEY-TEMPLATE NEWLINE
      NEWLINE
      "****************************************"

< SHOW SURVEY-TEMPLATE
Template:
   SURVEY-TEMPLATE (associated with) PROFILE
      MESSAGE-NO HANDLE Date: Subject:
```

2 A PRINT COMMAND CONTROLLED BY A TEMPLATE INCLUDED IN THE MESSAGE

Now the demonstration turns to Message No. 1, which carries its own template information in the message itself. We see the message printed using the PTEMPLATE. Next, the PTEMPLATE itself is displayed and finally the same message is printed through the template VERBATIM, which shows exactly what the message stored in the message-file looks like. The instructions field contains instructions that create a template. This template is extracted and used to print the message.

Demonstration No. 2.

```
< PRINT 1 CDEMO
##########------  Special Message! ------**********
This message is being transcribed thru a template in the message itself.

To:  lhermes
Subject:  The RFC
          Some comments by three people.

     I think that the RFC    Frank...You just        Looks good.  I'd
     which has been sent     don't know what         suggest that you
     around should be        you are talking         remove the
     adopted with no         about.                  restrictions
     modifications.          I strongly suggest      on who gets to use
                             that we get             the mechanism,
                             together tomorrow       however.
                             and talk about it.

< SHOW PTEMPLATE
Template:
   PTEMPLATE (associated with) PROFILE
      if Instructions: field exists
        then use extracted template from Instructions: field
        else VERBATIM
```

```
< PRINT 1 CDEMO VERBATIM
Date:   6-Sep-77 19:31:02-EDT
Sender:  VITTAL at BBN-TENEXD
Cola:  I think that the RFC
       which has been sent around should be adopted with no modifications.
Colb:  Frank...You just don't know what you are talking about.
       I strongly suggest that we get together tomorrow and talk about it.
Colc:  Looks good.  I'd suggest that you remove the restrictions
       on who gets to use the mechanism, however.
To:  lhermes
Subject:  The RFC
          Some comments by three people.
Instructions: ("#####  ...

This might be the body of the RFC, if we had one.


< SHOW AUTO-FORMAT
Template:
   AUTO-FORMAT (associated with) CDEMO (file: <LHERMES>CDEMO.TXT.1 )
      "##########******------  Special Message! ------******######" NEWLINE

      "This message is being transcribed thru a template in the message
 NEWLINE
      NEWLINE
      To:+ NEWLINE
      Subject:+ NEWLINE
      NEWLINE
      columnize field Cola: between 5 and 25
                field Colb: between 28 and 45
                field Colc: between 50 and 70
```

3 A PRINT COMMAND THAT TRIGGERS A QUESTIONNAIRE

Message No. 3 also contains its own template information, which consists of a set of instructions for creating and executing a questionnaire. The results of the questionnaire are packaged in a message which can be returned to the sender of Message No. 3. In the demonstration, the user runs the questionnaire twice, and ultimately refuses to send it at all.

Demonstration No. 3.

```
< PRINT 3 CDEMO
This is a questionnaire about baseball.
Do you want to answer it now? No
Thank you - please remember to respond to this questionnaire before Thurs

< PRINT 3 CDEMO
This is a questionnaire about baseball.
Do you want to answer it now? Yes
Do you follow baseball? No
Thank you very much...
**********************************************
A message is about to be sent.
What would you like to do with it?
one of:
Print it
Send it
Don't send it

What would you like to do with it? Print it

Q1: NO
To:  VITTAL at BBN-TENEXD

What would you like to do with it? Don't send it
```

```
< PRINT 3 CDEMO
This is a questionnaire about baseball.
Do you want to answer it now? Yes
Do you follow baseball? Yes
Which team do you root for? Yankees
How do you think the Yankees
     will do in the pennant race? 1st
Will they win the World Series? Yes
No way... Phillies in six.
Thank you very much...
**********************************************
A message is about to be sent.
What would you like to do with it? Print it

Q1: YES
Q2: Yankees
Q3: 1ST
Q4: YES
To:  VITTAL at BBN-TENEXD

What would you like to do with it? Don't send it
```

4 A MESSAGE CHOPPING OPERATION

Messages 4 through 7 illustrate a military chopping operation. Three different officers comment on, or "coordinate" a message, and the message is returned to its author with the comment display in three columns.

Demonstration No. 4

```
< COMPOSE TEMP-MSG CDEMO FOR.COORD
Subject: The bomb.
C1: able
C2: Cain
C3: Ross
Type text of message to ^Z:
This is a test.

John


< PRINT 4 CDEMO
Here is a message that  Cmdr. Nobody at BBNA  wants you to CHOP.

Subject:  The Atlantic Conflict
The message currently is:

It is clear that something should be done.
We recommend that no military action be taken.
But, we MUST insure peace; detente is the key.

                                        Cmdr Nobody
                                        COMTECPACF


Do you want to CHOP now? Yes
Type your comments to a ^Z:
C1's comment.

**********************************************
A message is about to be sent.
What would you like to do with it? Don't send it
```

```
< PRINT 5 CDEMO
Here is a message that   Cmdr. Nobody at BBNA  wants you to CHOP.

Subject:  The Atlantic Conflict
The message currently is:

It is clear that something should be done.
We recommend that no military action be taken.
But, we MUST insure peace; detente is the key.

                                        Cmdr Nobody
                                        COMTECPACF


CHOP comments by  Cmdr. Jones @ ISI

 I agree.  However, there must be a way to maintain detente.  \Joe

Do you want to CHOP now? No

NIL
< PRINT 6 CDEMO
Here is a message that    Cmdr. Nobody at BBNA  wants you to CHOP.

Subject:  The Atlantic Conflict
The message currently is:

It is clear that something should be done.
We recommend that no military action be taken.
But, we MUST insure peace; detente is the key.

                                        Cmdr Nobody
                                        COMTECPACF


Comments by other coordinators:
     Cmdr.  Jones @ ISI                 Col.  Smith @ BBNA

     I agree.  However, there must      OK.  \John
     be a way to maintain detente.
     \Joe

Do you want to CHOP now? No
```

```
< PRINT 7 CDEMO
Message returned from CHOP.
Subject:  CHOPping is completed.
          The Atlantic Conflict

Original message was:
It is clear that something should be done.
We recommend that no military action be taken.
But, we MUST insure peace; detente is the key.

                                        Cmdr Nobody
                                        COMTECPACF


Coordinators comments:
     Cmdr. Jones @ ISI     Col. Smith @ BBNA     Gen. Doe @ SRI-KA

     I agree. However,     OK. \John             I agree with Joe.
     there must be a way                         Mention something
     to maintain detente.                        about maintenance.
     \Joe                                        \Stan

< SHOW FINISHED-CHOPPING
Template:
   FINISHED-CHOPPING (associated with) CDEMO (file: <LHERMES>CDEMO.TXT.1 )
      "Message returned from CHOP." NEWLINE
      Subject:+ NEWLINE
      Suspense-date:+ NEWLINE
      "Original message was: " NEWLINE
      Text: NEWLINE
      "Coordinators comments:" NEWLINE
      columnize field C1-owner: between 5 and 24
                field C2-owner: between 27 and 46
                field C3-owner: between 49 and 68 NEWLINE
      NEWLINE
      columnize field C1-comments: between 5 and 24
                field C2-comments: between 27 and 46
                field C3-comments: between 49 and 68
```