U.S. Department
of Commerce

National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-68

The Expert Assistance System for the NBS Network Access Machine

Computer Science and Technology

NBS Special Publication 500-68

The Expert Assistance System for the NBS Network Access Machine

Shirley Ward Watkins

Center for Computer Systems Engineering
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE
Philip M. Klutznick, Secretary

Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation

National Bureau of Standards
Ernest Ambler, Director

Issued November 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer unitilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-68
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-68, 44 pages (Nov. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600178

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $2.50
(Add 25 percent for other than U.S. mailing)

TABLE OF CONTENTS

DISCLAIMER

Certain commercial products are identified in this special publication in order to adequately specify the Network Access Machine and the Expert Assistance System. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the products identified are the best available for the purposes described.

The Expert Assistance System For The NBS Network Access Machine

Shirley Ward Watkins

ABSTRACT

The Expert Assistance System (EAS) was developed at the National Bureau of Standards as a prototype to assist network users. Network users are faced with the problem of learning different procedures in order to access similar services on different host systems. A great deal of research has been precipitated by the desire to simplify network usage and many tools have been developed to assist the network user.

One of the approaches taken in network assistance has been to implement an intermediary machine. The intermediary machine translates simple user commands into the sequences of network and system commands required for execution on a target host system; thus, the user learns one set of commands which are applicable on different systems and networks. An ironic consequence of such an approach is that if the user desires to expand the basic set of functions provided by the intermediary machine or to tailor existing functions to individual needs, the user has to learn another command language -- that of the intermediary machine itself.

The EAS addresses the problem of building procedures for an intermediary machine. The EAS automatically generates procedures by recording an interaction between a user and network system and then translating this interaction into the commands required for execution on the intermediary machine. Development of the EAS was facilitated by the existence of an intermediary machine at the National Bureau of Standards -- the NBS Network Access Machine (NAM).

This report briefly describes the motivation for the development of a network assistance technique, discusses the design and implementation of the EAS at NBS, and then concludes with a view of future enhancements to the current EAS. The context for the description of the EAS is the NBS NAM; however the concepts are applicable to the general field of network user assistance.

Key words: Command languages; Communications; Computer access; Computer networks; Minicomputers; Protocols; User interfaces.

This Special Publication is one of a series prepared as part of a jointly sponsored effort by the National Bureau of Standards and the U. S. Air Force Rome Air Development Center under contract number F 30602-77-F-0066.

1.0 INTRODUCTION

The diversity of service offerings provided by computer networks was a major catalyst to the growth in popularity of networking. In addition to sparking the interest of computer enthusiasts, networking proved to be an attractive alternative to another class of computer user--those desiring access to remote computer resources or services. Services are defined as programs or combinations of programs provided by a computer system: e.g., assemblers, compilers, information retrieval systems, report generators and text processors. Due to this interest in the "end products" of networking, the members of this class are referred to as end users.

In attempting to use the services provided by heterogeneous networks, end users are faced with the problems of accessing services provided on different systems. Even the same services may have different access procedures when offered on different systems. In response to this access problem, numerous approaches to aiding end users have been investigated. As reported by Rosenthal [ROSER 76], these approaches fall into a number of categories. Some of these categories reflect the type of assistance provided (basic communications assistance, resource identification, resource connection, and service integration), while others reflect the architecture of the assistance technique (distributed assistance, minicomputer hosts, and intelligent terminals). An implementation of one approach is the National Bureau of Standards' Network Access Machine (NAM) in which user procedures are implemented as macros which drive a user/system session.

Following a general discussion of network assistance, this report will present details on the design and implementation of the NBS Expert Assistance System which is an aid to the NAM user. A brief description of the NAM is also included, with references to reports providing additional information.

2.0 NETWORK ASSISTANCE

Watkins and Kimbleton [WATKS 78] make several observations concerning network assistance which influenced the decision to extend the capabilities of the NBS NAM. Key points are summarized here; the reader is referred to the original report for justifications. Of the three major network user categories (end users, applications programmers, and systems programmers), the category which is most receptive to network access support is that of end users. Further, if a network access technique limits itself to support of this restricted audience, the development of the required access support still proves difficult. This

difficulty is due to the following user variations: user skill, rate of change in accessing services, usage intensity, and access complexity associated with a given service. Watkins and Kimbleton [WATKS 78] also observe that access to a service proceeds in four stages: acquisition, initialization, utilization and termination. Given these stages, it appears that centralized access support is not amenable to the utilization stage. The utilization stage requires intricate knowledge of the services being accessed; it is unreasonable to expect any individual or group of individuals to be knowledgeable in all network services. Therefore, such support should be provided either by user groups for the services, by the service developers, or through service access standardization.

These observations are confirmed when reviewed in relation to the NBS NAM. The major interest in using the NAM comes from the end users: e.g., librarians, information retrieval specialists. The stages of network access most generally addressed are the acquisition, initialization, and termination stages; when the utilization stage is supported, "experts" in given services have been called upon to formulate a library of support macros, one of the primary NAM mechanisms for executing user procedures.

There is another observation which may be made concerning usage of the NBS NAM; typically NAM users at some time wish to incorporate something new or different into the already existing macro library. At this point, they must write their own macros. Even if the end users restrict their macros to the minimally required NAM directives, there are complications; namely, detailed knowledge of the character strings (which may contain non-printing characters) received from a host system is required. Therefore, the end user would probably find it inconvenient, at best, to create individualized macros. Remembering that end users do not necessarily reflect a programming interest and that end users form the major category for access support, it seems reasonable that a technique be devised to aid them in the development of NAM macros.

In recognition of the problems encountered by the end users in attempting to create individual macros, a technique was designed to autmatically generate macros for them.

3.0 THE EXPERT ASSISTANCE SYSTEM

As noted above, NAM macros require exact specification of character strings which are to be received from host systems. If the specified character strings contain inaccurate or incomplete information, the NAM controlled interaction with the host system will not proceed properly.

In order to insure the inclusion of the proper character strings, a usual procedure is to use the NAM to connect to the target system and then enter the command strings which should be sent to the target system by the NAM. An option is available in the NAM to create a transcript of an interaction for the NAM user. This transcript provides a record of all characters received from the target system (including normally non-printing or control characters). Using this transcript, the user then generates the required NAM directives.

Based on the observation that a user/target system interaction precedes the creation of NAM directives, it was concluded that it would be appropriate to provide a special subsystem to generate these directives based on the transcript. This facility is called the Expert Assistance System (EAS).

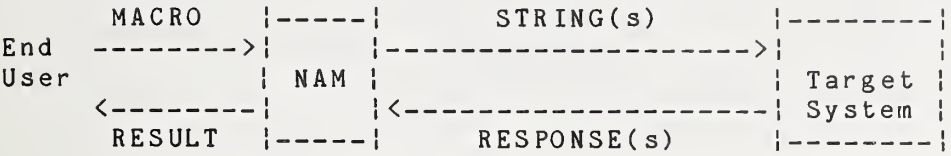
In this section, an overview of the NBS NAM will be provided followed by the details of the design of the NBS EAS. The NBS EAS will be described by presenting the design axioms, by providing a functional description of the EAS through a discussion of the increasing levels of EAS capability, and by describing the software components of the system.

3.1 The NBS Network Access Machine

The Network Access Machine (NAM) at the National Bureau of Standards facilitates access to network resources for the end user, [BLANR 74], [PYKET 73], [ROSER 74], [ROSER 76]. The NAM is implemented on a PDP 11/45 running the UNIX operating system. (UNIX is a trademark of Bell Laboratories.) Support is provided through three primary mechanisms: macros which support expansion of simple user-entered commands into the command sequences executable on specific networks and host computers connected to the network; a response analyzer allowing alternative responses (typically the expected response plus error conditions which could occur) during the expansion of a macro; and control mechanisms (case statements, if-then-else,...) which, collectively, constitute a command language level programming language. Figure 3.1-1 shows a simple interaction in which the end user enters the name of a macro; the macro is expanded into a sequence of NAM directives (some of which are character STRING(s) to be sent to the target system); and the response analyzer of the NAM inspects each RESPONSE which it receives from the target system to determine if any anomalies occur.

END USER/TARGET SYSTEM INTERACTION

Figure 3.1-1



legend:

MACRO	=	name of a macro
STRING(s)	=	a character string or series of character strings sent by the NAM to perform the function defined by the macro
RESPONSE(s)	=	the text strings sent by the target system in reaction to a command string sent by the NAM
RESULT	=	presentation of macro execution; RESULT may or may not be identical to the text string RESPONSE

The NAM design is based on the concept of presenting one uniform set of user commands which are executable across network boundaries and across heterogeneous host systems. In operation, user commands are first expanded into the command sequence appropriate to the system being accessed, responses are analyzed to determine if the interaction is proceeding as expected, anticipated errors are handled directly, and unanticipated errors are presented to the user for handling.

These capabilities have proven sufficiently powerful to support a number of applications. Rosenthal [ROSE2 78] discusses these applications which include: an aide to host connection ([BLANR 74], and [ROSER 74]); an experimental Remote Terminal Emulator ([ABRAM 77], and [MAMRS 78]); bibliographic information retrieval [TREUS 78]; and, "shopping" for network service based on the execution of benchmarks across target systems [ROSE1 78].

Each of the above applications had one key element in common -- the NAM macros which provide the foundation for each of these applications were written by persons quite versed in the NAM language. As previously mentioned, the NAM really provides a command language level programming language. This language requires that the user supply a minimum of the following information: the character string that should be sent to the target system, the conditions

which indicate that the character string received from the system in response to the user character string is completed (the termination conditions), and the character string which indicates that the response from the system is what was "expected" (the match conditions). If the full capabilities of the NAM are employed, the user can formulate quite sophisticated conditional statements, perform complex string operations, and in fact call any program which is available under the UNIX operating system.

3.2 Design Axioms

There are three design axioms which prevail in the EAS: (i) present this feature as an option to the user; (ii) keep the system compact; and (iii) construct an "easy to use" interface for the end user.

3.2.1 Optionality -

The EAS is implemented as an optional feature of the NBS NAM; it must be explicitly invoked by the end user. It was a design decision not to continually scan the end user/target system interaction to recognize repeated actions and to identify them to the end user. If the end user wants assistance in macro building, the EAS is invoked and the boundaries of the interaction to be converted into a macro are defined. The procedure is the following: the user issues a command to the NAM to invoke the EAS; the EAS records all characters exchanged between the user and the target system; the user issues a NAM command which terminates the recording. Following this recording procedure, the EAS translates the recorded interactions into the required NAM directives.

3.2.2 Compactness -

Compactness has been a primary consideration in the design of the EAS. The processing required by the current NAM software is considerable; therefore, extensions to that existing software must be carefully weighed to balance service provided and resource consumption. Since the EAS represents an attempt to minimize the frustration of the end user, it is imperative that the processing time of this added software not impact response time perceived by that user.

3.2.3 Ease Of Use -

Several features have been implemented to address the ease of use axiom: tailoring user/EAS interactions for two broad classes of end users, an on-line help feature, and added user response checking to buffer the end user from system diagnostics. Appendices A, B, C, D, and E present sample user sessions with the EAS to demonstrate the features designed in the interest of ease of use. These five appendices show the interactions required for translation of a transcript into a macro; all five use the same transcript.

There are two modes (normal and verbose) in which the end user may interact with the EAS.' These modes address two broad, experiential classes of EAS user. Normal mode (Appendix A) presumes that the end user is experienced with interacting with the EAS and as such is familiar with the questions asked and the procedures of macro translation. Verbose mode (Appendix B) is provided primarily for the novice or infrequent user of the EAS. By comparing Appendix A and Appendix B, the differences in these two modes are quite obvious. Verbose mode would probably be tedious once the end user becomes familiar with the EAS.

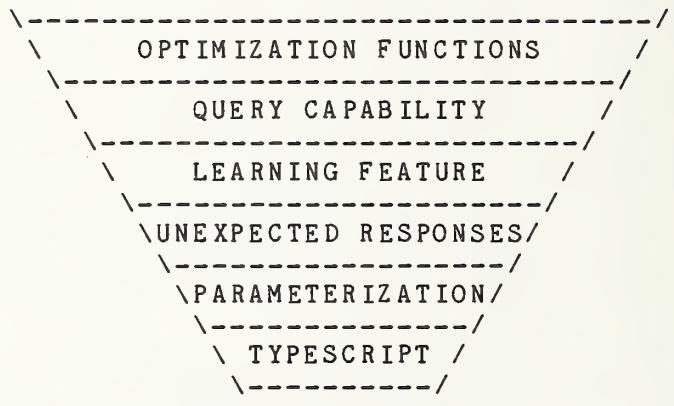
Appendices C and D demonstrate the on-line help facility of the EAS.' At any time an end user input is required, that end user may enter a "?" and receive all the options available for entry at that time. The EAS response to a "?" reflects the mode (normal or verbose) selected by the end user. Appendix C shows the help facility when normal mode is selected and Appendix D shows the same help features for verbose mode.

An attempt is also made to shield the end user from diagnostic messages orignating from the system on which the NAM and EAS are implemented. This is addressed by checking the responses entered by the end user for legality; thus, user responses containing information which is used for issuing operating system commands are screened for correctness. If the end user enters an unacceptable response (with respect to the operating system), the end user is informed of the viable options available at that point. An example of this type of assistance is entering the name of the macro to be created. This name also becomes the name of the file in which the macro is stored. If the format of the name is unacceptable, the EAS informs the user of the proper format. The EAS also interprets information provided by the operating system and presents it to the user in a meaningful way. An example of this type of assistance is again found in macro file naming. If the end user specifies a name which duplicates that of an existing macro, the EAS informs the end user and asks the user to verify that the existing file should be overwritten. Appendix E demonstrates these ease of use features.

3.3 Levels Of Expert Assistance

This section provides a functional description of the EAS. Following this level of description, the software components required to implement the EAS will be described.

Expert assistance may be subdivided into six increasing levels of sophistication: transcript, parameterization, unexpected response handling, learning feature, query system, and optimization. These levels form a hierarchy of assistance functions (see Figure 3.3-1). For the purposes of the NBS EAS effort, only the first three levels of assistance will be implemented. These levels are presented in this report by first giving a general overview of the capabilities implicit in a given level and then providing a description of the features of the NBS EAS at this level. Only a brief overview of the highest three levels is provided since they will not be incorporated into the NBS EAS.



Levels of Expert Assistance

Figure 3.3.1

3.3.1 Transcript -

3.3.1.1 Overview -

The minimum level of assistance provides a simple record/translate function. Based on the recorded user/system interaction, a sequence of NAM directives to replay a recorded interaction is produced. This level of expert assistance does not handle variations in a single system's response or minor changes in user commands, a support function that can be achieved through parameterization..

3.3.1.2 NBS EAS Implementation -

Initially, the utility of an EAS which only provided the transcript capability seemed non-existent. It was found that the need for parameterization was determined by the purpose of macro creation. There are two classes of EAS users: the individual end user and the "expert" user. The individual end user is characterized by an interest in building macros strictly for private use. The individual end user may not select parameterization for the acquisition, initialization, and termination stages of access because the macros are not intended for use by others.

The "expert" user is not an expert EAS user, rather an expert service specialist: for example, an expert in the use of specific bibliographic information retrieval systems, data base management systems, text processors, or compilers. The use of the EAS by the service specialist is characterized by building libraries of macros for a particular service. Note that this person is an expert on the target service, not on the EAS nor on the NAM. This is an important distinction; in past applications of the NBS NAM the macro libraries for services were built by a NAM expert in combination with a service specialist. For the purpose of building libraries of generally used macros, parameterization is required; for example, in the acquisition stage, user identifications and passwords required for login will vary.

3.3.2 Parameterization -

3.3.2.1 Overview -

At the parameterization level of expert assistance, certain fields of user commands can be identified as variable. Such identification can either be explicitly provided by the user or by the EAS based on its knowledge of commands and services. The identified fields can then be incorporated as parameters to the macro built by the EAS.

3.3.2.2 NBS EAS Implementation -

It is the intent of the design of the NBS EAS to implement parameterization by both allowing the user to force field identification and by providing a knowledge base of a few known systems and subsystems. However, in the current implementation, field identification is only provided through use of the knowledge base. As will be discussed in section 3.4, the developmental system has limited flexibility. User identification of fields is straightforward and discussion of how it can be incorporated into an operational model is provided here.

User specified parameterization would be designated via a special character. This character would by default be a specific control character; however, a command should be made available to redefine this special character to be any character convenient for the user. When the user enters this special character, it would be trapped by the recording module of the EAS and stored in the transcript file but not forwarded to the target system. If in fact the end user wished to send the special character to the target system, the character would be entered twice.

Figure 3.3-2 demonstrates the differences between what would be placed in the transcript file and what would be sent to the target system when the special character is used. The "@" designates the special character strictly for purposes of clarification for this example. In CASE 1, a field is flagged as variable; in CASE 2, the special character is sent to the target system.

Handling the Attention Character

Figure 3.3-2

	User Enters	Placed In Transcript	Target System Receives
CASE 1:	log @Watkins	log @Watkins	log Watkins
CASE 2:	log @@Watkins	log @@Watkins	log @Watkins

It is obvious that user identification of parameters is not consistent with the philosophy of providing an environment to build macros without imposing programming-type techniques on the end user. However, in order to provide a flexible system, this option should be included. As discussed above, expert users (service specialists) will require parameterization because they build libraries of macros for others. It is assumed that the service specialist is adequately versed in the target service to determine what types of information will change from user to user or session to session and should therefore be designated as parameters.

The knowledge base of the prototype NBS EAS (described in section 3.4.3) is small; however, the structure used to implement this base is sufficiently flexible and easy to augment that future enhancements could be conveniently made. Knowledge base augmentations to the prototype NBS EAS necessitate program changes and recompilation of the EAS code. However, as will be discussed in the enhancements section, an extension to the EAS could allow the end user to make additions to the knowledge base by entering an EAS subsystem which would query the user for needed information. The following is an example of how parameterization via a knowledge base would work: the EAS recognizes that the user is logging in to a system; if that system is known by the EAS, the EAS also knows what user identificationentifications (e.g, identification, password, account number) are required to login; based on this knowledge, the EAS knows what fields are candidates for parameterization.

3.3.3 Unexpected Response Handling -

3.3.3.1 Overview -

The unexpected response is certainly not an uncommon event in a networking environment. Therefore, macros should be flexible enough to recognize certain system responses, which do not match what was built into the macro as expected, and to deal with them properly. There are two approaches to handling the receipt of an unexpected response: a priori identification of all possible responses together with incorporation of appropriate actions; and user notification of the occurrence of an unexpected response which relies upon the user to specify the next action to be taken.

The a priori approach would be implemented via a preprogrammed EAS knowledge base. There are two impediments to this approach: identification of a large number of responses which are dependent upon such command parameters as the network, host system, and subsystem being used, and the remaining probability one-time or infrequent system responses (e.g., error or diagnostic messages) will still occur. It is certainly reasonable to implement a small knowledge base containing those "expected" unexpected responses which occur with some regularity (e.g., broadcast messages from the operators which are preceded by a constant tag). Therefore, coupling the preprogramming of a small base of such unexpected responses with user notification appears to be the most logical method of dealing with the unexpected response syndrome.

3.3.3.2 NBS EAS Implementation -

This level of expert assistance is not implemented in the current NBS EAS; however, the anticipated implementation will closely mirror the above overview and the design considerations presented in this section.

In order to determine which unexpected responses are reasonable for the EAS to handle, the reasons for their occurrence must be investigated in addition to the methodology for incorporating the procedures for handling unexpected responses. Basically, there are four reasons for receiving an unexpected response:

1. An operating system change in which system messages to the user community differ from the messages previously sent,
2. A one-time or rare occurrence of a particular system message,
3. A target system crashes, and
4. Responses which are not classified as "expected" but occur with some frequency.

The following methodology is used to incorporate unexpected response handling: during macro expansion if an unexpected response is encountered, the knowledge base of the EAS will be scanned for a match; if a corresponding entry is found, the EAS uses the preprogrammed procedure for that response; if the entry is not found, the EAS presents the received response to the end user and queries the user on how to proceed. There are two ways to proceed: abort the current expansion or have the user intercede followed by continuation of the macro expansion from the point at which the anomaly occurred. If the user selects the latter, the option also exists to incorporate the user procedures for handling the anomaly into the existing macro.

The NBS EAS builds macros which consist of terminate, match and send triplets. If the user wishes macros to be expanded to handle more than one expected response, this triplet must be expanded. Namely, conditional statements must be added to the macro to allow for multiple correct possibilities. Logically, the macro would resemble the following:

```
If Response == "Expected Response 1" then execute
    procedure 1
If Response == "Expected Response 2" then execute
    procedure 2
...
...
If Response == "Expected Response N" then execute
    procedure N
```

Now that the reasons for receiving unexpected responses have been given and the methodology for building procedures to handle unexpected responses has been described, the suitability of expecting the NBS EAS to handle unexpected responses will be investigated. In the case of an operating system change which eliminates the probability of receiving the previously used system responses, it is unreasonable to have the current macros extended; this process would perpetuate useless information. The user has two options for dealing with such a change: delete the previous macros and regenerate new ones via the EAS, or modify the existing macros via an editor. The former option does not presume user knowledge of the command language requirements of the intermediary machine, while the latter does.

In the case of a one-time or rare occurrence of a particular system response, it is unnecessary to extend the current macro. A more reasonable procedure would be for the user to abort the current expansion and try again, or to intercede with the target system and continue macro expansion from that point.

If the target system crashes, the current macro must be aborted. The user, of course, could then go to another target system and proceed, if the required services are available.

For handling responses that do occur with some regularity, it is certainly reasonable for the end user to expect assistance. The very common "expected" unexpected responses for services known to the EAS implementors would be contained in the EAS knowledge base. However, for those services or operating conditions not known to the EAS, the capability would exist for the end user to deal with the anomaly and have the resulting interaction(s) incorporated into the existing macro. It would be the decision of the end user whether or not the actions taken upon receipt of an anomaly would be built into a macro.

3.3.4 Upper Levels Of Expert Assistance -

The three upper levels of expert assistance are a learning feature, a query capability, and optimization. These levels are not be implemented in the NBS EAS; however, a brief description of each is provided in the interests of completeness.

3.3.4.1 Learning Feature -

The learning feature is similar to the type of capability provided by unexpected response handling: a knowledge base of services is used to enhance the macro generation process. However, in the learning feature the knowledge base of the EAS is dynamically expanded without user intervention. For example, based on one user's interaction with a service, a system message may be encountered for the first time; after observing the user/system interaction(s) triggered by this response (if any), the EAS automatically adds this unexpected response to its knowledge base; when this "expected" unexpected response occurs in a similar situation with another user, the EAS uses this new information.

It is anticipated that the type of information dynamically incorporated into the EAS knowledge base would no doubt take the form of system level messages which should

be ignored when encountered rather than generating faulty macro expansions. However, identifying the types of activity which are candidates for the learning feature requires thorough examination to determine inclusion. decided which are to be included. Certain activities are too general applicability individualized to assume; typical of the type of activity which would probably not be included is that which deals with the utilization stage of access (discussed in section 2.0).

The concept of incorporating a learning feature is not new; the RAND Corporation provided a similar feature in its prototype RITA system which acted as a user agent. Anderson [ANDE1 76] and [ANDE2 76] discusses the design of RITA.

3.3.4.2 Query Capability -

At the level of the query capability, the user is guided by the EAS through the entire process of macro generation. The user simply specifies the target system and objective of usage (general class of activity) and the EAS automatically generates the appropriate macro. If user-specific information is required for macro generation, the EAS queries the end user for the data.

This level of expert assistance requires an extensive knowledge base of systems and services. As such, it also would consume large amounts of computer resources to support the expert assistance software.

3.3.4.3 Optimization -

The highest level of expert assistance provides optimization support. At this level, the expert assistance system suggests to the user that certain activities are being repeated and could be incorporated into a macro. Further, the system suggests that more efficient ways to perform certain activities exist.

While these objectives are beyond the scope of the NBS EAS, and, in fact, are contrary to the NBS EAS design axiom of optionality, such support mechanisms have been considered for provision as part of a military message processing system (see [HEAFJ 74]).

3.4 Components Of The EAS

This section describes the software components required to implement the features of the EAS discussed previously.

The EAS consists of three major components: the recording module, the translation module, and a knowledge base. The translation module produces the NAM directives with information supplied by the recording module and the knowledge base. This section will describe the current implementation of these three components.

In its operational state the EAS would be a subsystem of the NAM; however, during its developmental stages, it was implemented independently of the NAM. As a result the recording module operates in a more isolated environment. For example, using the NAM a user may alternate between issuing commands to the NAM (in the form of macro calls or other directives) and to the target system directly. However, using the recording module of the EAS, it is not possible to enter characters directed only to the EAS. Everything entered by the user is sent to the target system.

3.4.1 The Recording Module -

The recording module of the EAS is quite simple: a target system is specified and a connection is established with that system. Once the connection is made, all characters exchanged between the target system and the end user are recorded. Figure 3.4-1 shows an interaction with the recording module, RECORD. RECORD is called with one parameter, the name of the system to which connection is to be made. The interaction shown in Figure 3.4-1 produced the transcript used for building the macros shown in Appendix G and Appendix H. The underlined portions of Figure 3.4-1 are the user-entered character strings.

Creation of the Transcript

Figure 3.4-1

```
% record isi[CR]
[LF][CR][LF]ISI-TENEX 1.34.40, ISI-SYSTEM-A EXEC
1.54.27[CR]
[LF]@login sww passwd [CR]
[LF]JOB 36 on TTY36 8-Nov-79 08:43[CR]
[LF]PREVIOUS LOGIN: 1-NOV-79 09:42[CR]
[LF]@dir[CR]
[LF][CR]
[LF]<SWW>[CR]
[LF]]ARCHIVE-DIRECTORY[.;1[CR]
[LF]@logo[CR]
[LF]KILLED JOB 36, USER SWW, ACCT GEN-NBS, TTY 36,
AT 11/08/79 0845[CR]
[LF]USED 0:0:2 IN 0:0:27
```

User and system characters (including system echos) are recorded using thep83300 exact time sequence they occur; the time at which characters occur is not recorded but the order of occurrence is preserved. In order to provide the translation module with an indication of which characters were originated by the user and which by the system, the recording module must record a flag for the post processing of the recorded data. Character information requires 7 bits for storage and 8 bits (one-half of a PDP 11/45 word) are provided. Therefore, the eighth bit of every character is available. This eighth bit is used to indicate if the character is user or system generated. If the eighth bit is set (equals 1), the character is from the system; if the eighth bit is not set (equals 0), the character is from the user.

3.4.2 The Translation Module -

Once the user terminates the recording of a user/system interaction, the translation process begins. While the overall responsibility of the translation module is to create macro directives, implicit in this function is echo removal and parameterization.

The translation module requests certain information from the user which is essential to the macro generation process. Appendices A through E are samples of user interactions with the translation module. The information requested by the translation module includes: the name of the macro to be built, the name of the system on which the macro will execute, and if the system and user activity are

recognized, a user decision concerning parameterization. The first piece of information, the name of the macro, allows the user the flexibility to invoke macros by names which are meaningful to the user rather than having the translation module create names in some homogenized fashion. The name of the system is required as a pointer into the knowledge base which contains the known services and the potential parameters. The last piece of information provides a parameterization option.

3.4.2.1 Macro Directives -

The translation module builds triplets of NAM directives in the sequence: .term, .match, and .send. In the .term directive the conditions which must be satisfied in order for a system response to be judged complete are specified. These conditions may take one of two forms: a timeout or a character string. A timeout is the maximum number of seconds which may occur between the last user character transmitted and the first system response character received or between two consecutive system characters. In choosing a character string to be used for a terminate condition, the shortest, unique string is normally the best candidate. The requirement for uniqueness is mandatory: as soon as the terminate character string is encountered, the NAM ignores the remainder of the system characters sent and continues processing the macro. The possibility exists that an incomplete system response will result in the .match condition being unsatisfied or the next user string being prematurely sent to the target system.

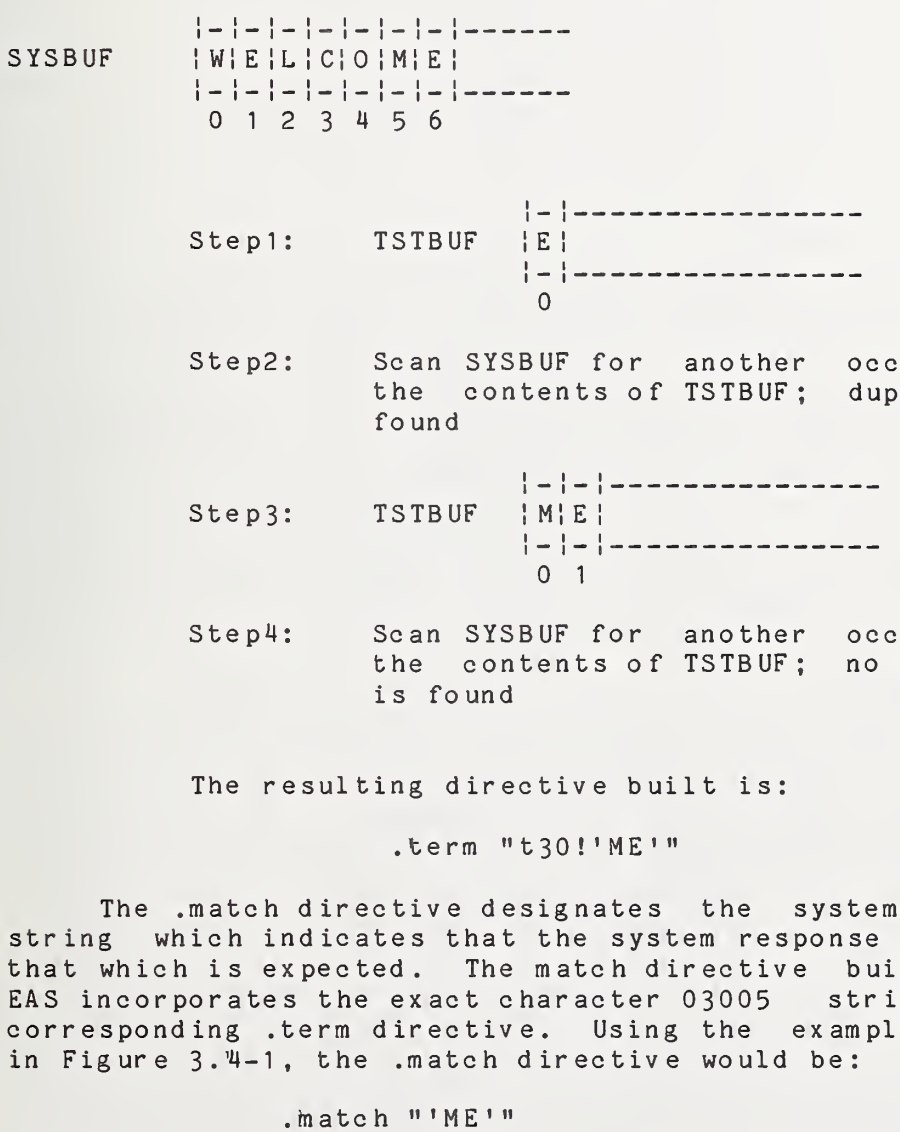
The translation module builds terminate conditions which are satisfied by a timeout or a character string. The timeout was arbitrarily chosen to be 30 seconds; it was felt that this interval insured that a timeout would be the result of a target system crash and not simply slow response due to system loading.

The user/system interactions are scanned in couplets of transactions; that is, the file built by the recording module is viewed by the translation module as pairs of user initiated stimuli and their associated system responses. In this context, the terminate character string is built by scanning a particular system response beginning with the last character sent by the target system. The current system response is placed in the buffer SYSBUF. The last character is considered a candidate as the terminate string; as such it is placed in the buffer TSTBUF for testing. The remainder of the system response is scanned for the reoccurence of this character; if the character is not found, the terminate string consists of a timeout of 30 seconds or an occurence of that one character; if the character is found, then that character is moved to the next

position in TSTBUF and the character preceding that character in the system response becomes the first character in TSTBUF. At this time the remainder of the system response is scanned for the re-occurrence of these two characters and the process is repeated until a unique character string is found. Figure 3.4-1 demonstrates the process of finding a unique character string.

Building A Unique Character String

Figure 3.4-1



The resulting directive built is:

```
.term "t30!'ME'"
```

The .match directive designates the system character string which indicates that the system response received is that which is expected. The match directive built by the EAS incorporates the exact character 03005 string as the corresponding .term directive. Using the example provided in Figure 3.4-1, the .match directive would be:

```
.match "'ME'"
```

The .send directive designates the user character string to be transmitted to the target system. A sample .send directive is:

```
.send "Login Watkins[CR]"
```

Every character occurring within the double quotes is transmitted to the target system. The exception is marked by encountering square brackets. Bracketed terms indicate non-printing or control characters from the ASCII character set which should be transmitted. Appendix H lists these abbreviations. When the user desires parameterization, the character string which is variable is denoted by "$N" where N is a positive integer. Using the above login request, the following would be the .send directive generated:

```
.send "Login $1[CR]"
```

Parameterization is discussed in more detail below.

3.4.2.2 Echo Removal -

The recording module maintains a record of all characters exchanged between the user and target system; it is the responsibilty of the translation module to distinguish between systems characters which are part of a system response and those which are echos. In a full duplex system, characters entered by the user are sent back to the user's terminal (echoed) after they are received. If the record containing echos is used to build the macro directives, there would be a .term, .match, and .send triplet built for every character entered by the user. Therefore, echos must be stripped from the recorded interactions prior to macro generation.

The algorithm for echo removal produces an approximation for the half duplex representation of a full-duplex interaction. While the sequence of characters is preserved in an echo, the timing is not. Any number of user characters may be interposed between a user character and its echo. The problem is to identify when a character transmitted from the target system is an echo and when it is the beginning of a sequence of system response characters. The algorithm requires that the beginning of a user character sequence be identified. This is accomplished by defining any character from the user as terminating a system transmission. Likewise, any character from the system which is not an echo terminates the user transmission. The procedure involves placing user characters in a buffer (USRBUF), and maintaining pointers to the end of USRBUF and to the current user character (CHR). Each target system character received is an echo candidate and has to be compared with the current user character. As long as a

match exists, the pointer is advanced and the process repeated. When the end of the buffer is reached, the user transmission is terminated. If there is a nonmatch before the end of the buffer, the remaining user characters end the target system transmission and begin the next user transmission, irrespective of the time at which the various characters occurred.

There is one exception for a nonexact match. An upper case character from the target system is treated as a match to the corresponding lower case character from the user. This option is required by the existence of target systems echoing all uppercase characters irrespective of the case of the user character received.

3.4.2.3 Parameterization -

Parameterization is provided as an option to the user and is implemented via a knowledge base. The parameterization algorithm is dependent upon the following data: the name of the system on which the macro is to execute, the relative position of the user message, the contents of the user message, and the user's parameterization decision.

The name of the system is used as the main pointer into the knowledge base of known systems and subsystems called "systparameter". Details of the structure of the knowledge base are provided in the next section. If the system name is not known (that is, there is no entry in the knowledge base), the variable KNWNFLG is set to a minus one (-1). The parameterization procedure is never invoked under the condition that "KNWNFLG = -1". If the system is found in the knowledge base (that is, KNWNFLG = 0), then the macro is a candidate for parameterization.

The next step in the parameterization procedure is dependent upon the sequential position of the user message in the transcript. For the purpose of the translation algorithm, a user message is viewed as being composed of individual words (character strings separated by spaces or carriage return). The first user message has significance in terms of the parameterization procedure because it is the one used to determine if potential parameters exist.

The contents of the first user message are scanned to determine if the user activity is defined in the knowledge base. The name of the system provides a pointer into the structure which contains the names of known activities (or subsystems) called "sub". The contents of the first line are compared with the names of subsystems to determine if a match exists.

If a match is found, the user is informed that the macro may be constructed with parameters, and what those parameters are. For example, if the system is recognized and the activity recognized is "login", the user would be informed of the following:

> "This macro may be built with the following parameters:
>
> id
> password
> account
>
> Should the macro be built with these parameters? "

If the user responds in the negative, then the character strings generated for the .send directives are identical to what was recorded in the transcript. If the user responds in the positive, then the character string corresponding to the first parameter is replaced with a "$1", the second with a "$2", etc. In the above example, the identification entered by the user would be replaced with $1, the password by $2, and the account by $3. To further clarify, the terms identification, password and account describe the type of information which would be entered as arguments to the macro call, not the character strings entered by the user. For example, the user-entered string for an identification may be "Watkins" therefore the string "Watkins" would be replaced by a $1 wherever it occurred in the macro.

In the current version, the user must accept all the parameters listed by the EAS or none. As described in the enhancements section, any combination of the potential parameters could be specified rather than an "all or nothing at all" condition.

3.4.3 The Knowledge Base -

The knowledge base contains system-specific and service-specific information used by the translation module of the EAS. At the current time the information contained in the knowledge base relates only to parameteriztion; however, the same structure defined in this section is suitable for unexpected response handling.

The language in which the NAM and EAS software is written is "C" which is a general-purpose programming language providing fundamental flow of control constructs and flexible structures [KERNB 78]. The structures provided by C made the implementation of the knowledge base straightforward, flexible and expandable.

A structure is a collection of one or more variables, possibly of different types, grouped together under a single name. A structure is declared using the keyword "struct" which is a list of declarations enclosed in braces. The structure may optionally be named by suppling a structure tag.

As described in section 3.4.2.3, the knowledge base of known systems used for parameterization is called "systparameter". Three systems were selected for inclusion in the experimental system: one at Bolt, Beranek, and Newman in Cambridge, Massachusetts (bbn), one of the Information Sciences Institute of the University of Southern California (isi), and one at the Experimental Computer Facility at the National Bureau of Standards (nbs10). EAS makes connections to all three via a packet-switching communications network (ARPANET) and to the NBS facility also via direct-dial.

The following is the structure declaration for the structure "systparameter":

```
struct systparameter{
        char *systname;
        struct sub *subsystpnt;
}systems[]{
            "nbs10",
              &nbssubsys,
            "bbn",
              &bbnsubsys,
            "isi",
              &isisubsys,
            0
};
```

where the name of the structure (structure tag) is "systparameter", and the members are pairs of character arrays (names of systems) and pointers to structures defined by a structure declaration named "sub". The "systems[]" label following the structure declaration "systparameter" defines an array "systems" of structures of this type. Each element of the array is a structure; by leaving [] empty the compiler computes the number of entries in "systems" thus allowing the expansion of the "systems" by simply adding a name/structure pointer pair and not worrying about maintaining prespecified array sizes.

The user supplies the name of the system which is compared to the names of systems contained in "systems" (nbs10, bbn, and isi). If a match is found, then the pointer to the appropriate structure with a structure tag of "sub" is used to determine if the subsystem or user activity is known.

The following is the structure declaration for "sub":

```
struct sub{
        char *subs;
        char **pntr;
}nbssubsys[]{
        "login",
         &tenloginparameters,
        "logout",
         &tenlogoutparameters,
        "mail",
         &tenmailparameters,
        0
},bbnsubsys[]{
        "login",
         &tnxloginparameters,
        "logout",
         &tnxlogoutparameters,
        "mail",
         &tnxmailparameters,
        0
},isisubsys[]{
        "login",
         &tnxloginparameters,
        "logout",
         &tnxlogoutparameters,
        "mail",
         &tnxmailparameters,
        0
};
```

The structures declared by a "struct" statement do not have to have the same number of member pairs; that is, using the above "sub" declaration as an example, login may be a known activity for all known systems, but mail may only be known for one of the systems.

To complete a description of the knowledge base the two variables "tenloginparameters" and "tnxloginparameters" follow:

```
char *tenloginparameters[]{
     "id",
     0
};


char *tnxloginparameters[]{
     "id",
     "password",
     "account",
     0
};
```

The knowledge base for unexpected response handling would be quite similar. The entry into the knowledge base would again be the name of the system on which the macro is running. Associated with each name, would be an array of pairs of "expected" unexpected system messages and the macro directives which should be associated with that message.

4.0 ENHANCEMENTS TO THE CURRENT NBS EAS

Three enhancements to the NBS EAS were identified in the development of the prototype. They are (in prioritized order): user selection of potential parameters, user-specified parameters, and a subsystem to expand the EAS knowledge base. The main factors used in this priority scheme were ease of implementation and immediate need.

4.1 User Selection Of Parameters

The EAS requires that the user accept all or none of the EAS-identified parameters for a macro. It is a straightforward task to query the user on each parameter and allow the user to select any combination of the parameters.

While the need for this option was not recognized in the design of the prototype system, it was during the trial usage. An example of the desirability of this feature was found in one of the currently known subsystems "login": a user may have one identification and account number for a system for the entire length of time that system is used; however, the user may change passwords frequently. Therefore, that user would probably desire to have the macro built with the identification and account as macro parameters and not the password.

4.2 User-specified Parameters

As identified in the section on parameterization, there are two ways to provide this feature: allow the user to flag a field as variable and use a knowledge base. Although the latter is the more complicated to implement, it is the only way currently provided to support parameterization.

The reason that the user can not flag fields is due to the prototype EAS configuration in which all characters entered by a user are directly sent to the target system. Had the system proceeded to an operational state, it would have used the NAM software for connecting to target systems, and the recording module could have trapped special characters.

The methodology in the operational EAS would be: a special character indicates that the user is flagging the following character string as variable; the special character is retained in the transcript file but not forwarded to the target system; if the user wishes to send that special character to the system, the user enters the special character twice; two of the characters are entered in the transcript file, but only one is sent to the target system. When the translation module processes the transcript file, encountering one special character indicates that "$N" (where N is a positive integer) should replace the special character and the character string following it in the .send directive; if two of these characters appear consecutively, one is placed in the .send directive.

4.3 EAS Subsystem To Expand The Knowledge Base

This enhancement would require the greatest amount of developmental effort and would in fact be available only to a few "privileged" users. The reason for this limitation is that the result of this activity is a recompilation of the EAS code.

This subsystem would enable a user to interact with the EAS to expand the knowledge base. The user would be guided through this activity by questions from the EAS. In this manner, new systems could be added to the knowledge base, or subsystems of known systems expanded. It was also envisioned that the knowledge base of "expected" unexpected responses could be expanded in this manner; however, if the user wished to expand this knowledge base, that user would have to be prepared to enter highly system-specific information. The user would have to supply what the exact message from the target system is which constitutes the "expected" unexpected response, what user message should be sent (if any) in response, and the target system response

which is to be expected in reaction to the new user message.

5.0 SUMMARY

This report has described an extension to the NBS Network Access Machine (NAM) which was implemented in 1978; however, the concepts are applicable to the general field of network user assistance. The report provides an introduction to the field of network assistance, a description of the NBS work in network assistance, and a technical description of the extension to the NAM called an Expert Assistance System (EAS).

The NBS EAS is a tool for users of the NBS NAM; it automatically generates macro directives for users of the NAM by observing a user/target system interaction and then translating this record into a macro suitable for execution on the NAM. It was designed as a strictly optional tool to relieve the NAM user of the more tedious tasks in writing macros.

The macros built by the NBS EAS utilize only the directives required by the NAM for macro expansion. There are some quite sophisticated commands for flow of control constructs and string matching which are not used. It was decided to focus on the smallest, feasible command set due to the design axioms of optionality, compactness, and ease of use. The prototype system has demonstrated that limiting the scope to these few required directives provides a sufficiently powerful system for building short macros for the acquisition, initialization, and termination stages of service access.

6.0 REFERENCES

[ABRAM 77] Abrams, Marshall D., "Techniques for Evaluating the Effectiveness of Interactive Computer Service", Proceedings 1977 ACM Annual Conference, pp. 452 - 458, 1977.

[ANDE1 76] Anderson, Robert H., and Gillogly, James J., The RAND Intelligent Terminal Agent (RITA), Rand Report R-1809-ARPA, February 1976.

[ANDE2 76] Anderson, Robert H., and Gillogly, James J., "The RAND Intelligent Agent (RITA) as a Network Access Aid," Proceedings of the National Computer Conference 1976, pp. 501 - 509.

[BLANR 74] Blanc, Robert P., "Assisting Network Users with a Network Access Machine," Proceedings of the ACM, 1974, November, 1974.

[FITZM 78] Fitzgerald, M. L., Common Command Language for File Manipulation and Network Job Execution: An Example, NBS Special Publicaton 500-37, August 1978.

[HEAFJ 74] Heafner, John F., A Methodology for Selecting and Refining Man-Computer Languages to Improve Users' Performance, Information Sciences Institute, University of Southern California, ISI/RR-74-21, ARPA Order No. 2223, September 1974, 53p.

[KERNB 78] Kernighan, Brian W., and Ritchie, Dennis M., The C Programming Language, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978, 228p.

[KIMBS 78] Kimbleton, Stephen R., Wood, Helen M., and Fitzgerald, M. L., "Network Operating Systems -- An Implementation Approach", pp. 773 - 778, Proceedings of the 1978 National Computer Conference, Volume 47, June, 1978.

[MAMRS 78] Mamrak, Sandra A., and Amer, Paul D., A Methodology for the Selection of Interactive Computer Services, NBS Special Publication 500-44.

[PYKET 73] Pyke, Thomas N., Jr., "Some Technical Considerations for Improved Service to Computer Network Users," Proceedings COMPCON 73, February 1973, pp. 53-55.

[ROSE1 78] Rosenthal, Robert, and Lucas, Bruce D., The Design and Implementation of the National Bureau of Standards' Network Access Machine (NAM), NBS Special Publication 500-35, June 1978.

[ROSE2 78] Rosenthal, Robert, "A Report on Several Experiments that Utilize the NBS NAM", internal memorandum, July 1978

[ROSER 74] Rosenthal, Robert, and Watkins, Shirley Ward, "Automated Access to Network Resources -- A Network Access Machine", Proceedings Computer Networks: Trends and Applications 1974, IEEE Inc., New York, 1974, pp. 47 - 50.

[ROSER 76] Rosenthal, Robert, A Review of Network Access Techniques with a Case Study: The Network Access Machine, NBS Technical Report 917, July 1976.

[TREUS 79] Treu, Seigfried, A Testbed for Providing Uniformity to User-Computer Interactive Languages, NBS Special Publication (in preparation).

[WATKS 78] Watkins, Shirley Ward, and Kimbleton, Stephen R., "Network Access Technology -- A Perspective", Proceedings of the National Computer Conference 1978, pp. 495 - 503, June 1978.

APPENDIX A

EAS INTERACTION IN NORMAL MODE

```
Do you wish to receive verbose or brief prompts?
    For verbose, type a v followed by a carriage return.
    For brief, type a b followed by a carriage return.
    For a fuller explanation, type ?  followed by a carriage
    return.
b
Macro filename?  logintoisil
System name?
isi

This macro may be built with the following parameters:
id
password
account

Parametization? (y/n)
y
%
```

APPENDIX B

EAS INTERACTION IN VERBOSE MODE

```
Do you wish to receive verbose or brief prompts?
     For verbose, type a v followed by a carriage return.
     For brief, type a b followed by a carriage return.
     For a fuller explanation, type ? followed by a carriage
     return.
v

Enter the name of the file (maximum of 14 characters)  which
will contain the macro followed by a carriage return.
logintoisi2


Name of system to which macro will be applied?  Enter  a  ?
followed  by a carriage return for a list of systems know to
the EAS.'
isi

This macro may be built with the following parameters:
id
password
account

Should macro be generated with these  parameters  marked  as
arguments  to  macro  invocation?  If  parameterization  is
desired, whenever this macro is invoked  arguments  must  be
provided.  Enter  y  for  yes  and  n  for no followed by a
carriage return.
n
%
```

APPENDIX C

ON-LINE HELP FACILITY IN NORMAL MODE

```
Do you wish to receive verbose or brief prompts?
    For verbose, type a v followed by a carriage return.
    For brief, type a b followed by a carriage return.
    For a fuller explanation, type ? followed by a carriage
    return.
?
```

In verbose mode, the prompts from the Expert Assistance System (EAS) contain much greater detail. In brief mode, it is assumed that the user of the EAS is familiar enough with the required inputs that minimal prompts are needed. Brief mode assumes prior usage of the EAS.' At any time an input is requested, the user may enter a ? followed by a carriage return. The following are your current options:

```
    v or V for verbose
    b or B for brief, or
    ? for this message.

Do you want verbose or brief prompts?
b
Macro filename?  ?
```

A filename may contain up to 14 printing characters (alphabetics, numerics, and special characters). Enter filename.

```
logintoisi3
System name?
?
nbs10
bbn
isi
System name?
isi

This macro may be built with the following parameters:
id
password
account

Parameterization? (y/n)
?
```

```
Parameterization is provided for subsystems known to the
EAS.'  If  you answer yes, then ALL of the parameters listed
below must be provided as arguments to macro invocation.  If
you answer no, then a macro will be built which will execute
a dialogue identical to the one just recorded.

This macro may be built with the following parameters:
id
password
account

Parametization? (y/n)
n
%
```

APPENDIX D

ON-LINE HELP FACILITY IN VERBOSE MODE

```
Do you wish to receive verbose or brief prompts?
     For verbose, type a v followed by a carriage return.
     For brief, type a b followed by a carriage return.
     For a fuller explanation, type ? followed by a carriage
     return.
v

Enter the name of the file (maximum of 14 charactrs) which
will contain the macro followed by a carriage return.
?

A filename by contain up to 14 printing characters
(alphabetics, numberics, and special characters). Enter
filename.

logintoisi4


Name of system to which macro will be applied? Enter a ?
followed by a carriage return for a list of systems know to
the EAS.'
?
nbs10
bbn
isi


Name of system to which macro will be applied? Enter a ?
followed by a carriage return for a list of systems known to
the EAS.'
isi

This macro may be built with the following parameters:
id
password
account

Should macro be generated with these parameters marked as
arguments to macro invocation? If parameterization is
desired, whenever this macro is invoked arguments must be
provided. Enter y for yes and n for no followed by a
carriage return.
?

Parameterization is provided for subsystems known to the
EAS.' If you answer yes, then ALL of the parameters listed
below must be provided as arguments to macro invocation. If
you answer no, then a macro wil be built which will execute
a dialogue identical to the one just recorded.
```

```
This macro may be built with the following parameters:
id
password
account

Should macro be generated with these parameters marked as
arguments to macro invoction? If parameterization is
desired, whenever this macro is invoked arguments must be
provided. Enter y for yes and n for no followed by a
carriage return.
y
%
```

APPENDIX E

USER'S PERSPECTION OF EAS HANDLING OF DIAGNOSTICS

Do you wish to receive verbose or brief prompts?
For verbose, type a v followed by a carriage return. For brief, type a b followed by a carriage return. For a fuller explanation, type ? followed by a carriage return.
v

Enter the name of the file (maximum of 14 characters) which will contain the macro followed by a carriage return.
loginlogintoisi

FILENAME TOO LONG!! A maximum of 14 characters is allowed. Reenter filename.

logintoisil

File ALREADY exists!!
Should the existing file be overwritten? Enter Y for yes and N for no followed by a carriage return. If you overwrite the file, the current contents of the file will be lost!
n
Enter filename
logintoisi5

Name of system to which macro will be applied? Enter a ? followed by a carriage return for a list of systems known to the EAS.'
isi

This macro may be built with the following parameters:
id
password
account

Should macro be generated with these parameters marked as arguments to macro invocation? If parameterization is desired, whenever this macro is invoked arguments must be provided. Enter y for yes and n for no followed by a carriage return.

APPENDIX F

ASCII ABREVIATIONS FOR NON-PRINTING AND CONTROL CHARACTERS

Abbreviation	Function	Octal Code
NUL	null character	000
SOH	start of header	001
STX	start of text	002
ETX	end of text	003
EOT	end of transmission	004
ENQ	enquiry	005
ACK	acknowlege	006
BEL	bell	007
BS	backspace	010
HT	horizontal tabulation	011
LF	line feed	012
VT	vertical tabulation	013
FF	form feed	014
CR	carriage return	015
SO	shift out	016
SI	shift in	017
DLE	data link escape	020
DC1	device control 1	021
DC2	device control 2	022
DC3	device control 3	023
DC4	device control 4	024
NAK	negative acknowledge	025
SYN	synchronous idle	026
ETB	end of transmission block	027
CAN	cancel	030
EM	end of medium	031
SUB	substitute	032
ESC	escape	033
FS	file separator	034
GS	group separator	035
RS	record separator	036
US	unit separator	037
DEL	delete	177

APPENDIX G

NAM MACRO CREATED BY EAS WITHOUT PARAMETERIZATION

```
cat logintoisi2
.term "t30 ! '@'"
.match "'@'"
.send "login sww watt  [CR][LF]"
.term "t30 ! '@'"
.match "'@'"
.send "dir[CR][LF]"
.term "30 ! 'IN '"
.match "'IN '"
.send "logo[CR][LF]"
%
```

APPENDIX H

NAM MACRO CREATED BY EAS WITH PARAMETERIZATION

```
% cat logintoisil
.term "t30 ! '@'"
.match "'@'"
.send "login "$1" "$2" "$3" [CR][LF]"
.term "t30 ! '@'"
.match "'@'"
.send "dir [CR][LF]"
.term "t30 ! 'IN '"
.match "'IN '"
.send "logo[CR][LF]"
%
```

*U.S. GOVERNMENT PRINTING OFFICE: 1980-0-340-997/313

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS SP 500-68	2. Performing Organ. Report No.	3. Publication Date November 1980

4. TITLE AND SUBTITLE

The Expert Assistance System for the NBS Network Access Machine

5. AUTHOR(S)

Shirley Ward Watkins

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as No. 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 80-600178

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The Expert Assistance System (EAS) was developed at the National Bureau of Standards as a prototype to assist network users. Network users are faced with the problem of learning different procedures in order to access similar services on different host systems. A great deal of research has been precipitated by the desire to simplify network usage and many tools have been developed to assist the network user.

One of the approaches taken in network assistance has been to implement an intermediary machine. The intermediary machine translates simple user commands into the sequences of network and system commands required for execution on a target host system; thus, the user learns one set of commands which are applicable on different systems and networks. An ironic consequence of such an approach is that if the user desires to expand the basic set of functions provided by the intermediary machine or to tailor existing functions to individual needs, the user has to learn another command language--that of the intermediary machine itself.

The EAS addresses the problem of building procedures for an intermediary machine. The EAS automatically generates procedures by recording an interaction between a user and network system and then translating this interaction into the commands required for execution on the intermediary machine. Development of the EAS was facilitated by the existence of an intermediary machine at the National Bureau of Standards--the NBS Network Access Machine (NAM).

This report briefly describes the motivation for the development of a network assistance technique, discusses the design and implementation of the EAS at NBS, and then concludes with a view of future enhancements to the current EAS. The context for the description of the EAS is the NBS NAM; however the concepts are applicable to the general field of network user assistance.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Command languages; communications; computer access; computer networks; minicomputers; protocols; user interfaces.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

44

15. Price

$2.50

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)