<REFERENCE>(VAX_NOTES_ CALL_ROUT_RM)

Order Number: DRAFT-VERSION DRAFT

September 1990

This draft is for freewar unsupptd distribution. Comments are welcome; please send to email::removed.

Revision/Update Information: This is a revised manual.

Operating System and Version: VMS Version 5.3 or higher.

Software Version: VAX Notes Version 2.1.

First Printing, March 1986
Revised, September 1990

© Digital Equipment Corporation 1986, 1990.

All Rights Reserved.

ZK_ _ _

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

Preface v

Part I VAX Notes Callable Routines and Network Protocol

1 VAX Notes Network Protocol

1.1 Type Field 1–1
1.2 Length Field 1–3
1.3 Value Field 1–4
1.4 Structure of a Data Packet 1–4
1.4.1 Example: Modifying a User/Membership Record 1–5
1.4.2 Type Numbers of Messages and Routines 1–7
1.5 Selected Item Codes and Descriptions 1–8
1.5.1 Protocol-Specific Item Codes 1–8
1.5.2 GET Item Codes 1–11

Part II VAX Notes Routines Dictionary

1 General Information 3
1.1 Attributes of Routine Arguments in VMS and ULTRIX 3
1.2 Categories of Routines 4
2 General Rules for Routine Formats 4
2.1 Valid Item Codes for Output Item Lists 5
NOTES$CLASS_ADD 8
NOTES$CLASS_BEGIN 10
NOTES$CLASS_DELETE 12
NOTES$CLASS_END 14
NOTES$CLASS_GET 16
NOTES$CLASS_GET_ENTRY 18
NOTES$CLASS_MODIFY 20
NOTES$ENTRY_ADD 22

NOTES$ENTRY_BEGIN 24
NOTES$ENTRY_DELETE 26
NOTES$ENTRY_END 28
NOTES$ENTRY_GET 30
NOTES$ENTRY_GET_CLASS 32
NOTES$ENTRY_GET_KEYWORD 34
NOTES$ENTRY_MODIFY 36
NOTES$ENTRY_UPDATE 38
NOTES$KEYWORD_ADD 40
NOTES$KEYWORD_BEGIN 42
NOTES$KEYWORD_DELETE 44
NOTES$KEYWORD_END 46
NOTES$KEYWORD_GET 48
NOTES$KEYWORD_GET_NOTE 50
NOTES$KEYWORD_MODIFY 52
NOTES$NOTE_ADD 54
NOTES$NOTE_ADD_TEXT 56
NOTES$NOTE_BEGIN 58
NOTES$NOTE_DELETE 60
NOTES$NOTE_END 62
NOTES$NOTE_GET 64
NOTES$NOTE_GET_KEYWORD 66
NOTES$NOTE_GET_TEXT 68
NOTES$NOTE_MODIFY 70
NOTES$NOTEFILE_BEGIN 72
NOTES$NOTEFILE_GET_INFO 74
NOTES$NOTEFILE_LIST 76
NOTES$NOTEFILE_LIST_BEGIN 78
NOTES$NOTEFILE_MODIFY 80
NOTES$NOTES 82
NOTES$PROFILE_BEGIN 84
NOTES$PROFILE_END 86
NOTES$PROFILE_GET 88
NOTES$PROFILE_MODIFY 90
NOTES$USER_ADD 92
NOTES$USER_BEGIN 94
NOTES$USER_DELETE 96

NOTES$USER_END 98
NOTES$USER_GET 100
NOTES$USER_MODIFY 102

Part III Appendixes

A Item List Format and Contents

A.1 Various Context Items A–1
A.2 Item Codes A–2

B Sample Program

C Messages

Index

Examples

B–1 Sample BASIC Program B–3

Tables

1 Attributes in VMS of Arguments to Notes Routines 3
2 Attributes in ULTRIX of Arguments to Notes Routines 4
A–1 Item Codes A–2

Preface

The routines described in this manual allow you to develop a new user interface for VAX Notes or to call VAX Notes from an application program.

Intended Audience

This manual is intended for programmers who will be writing a user interface for VAX Notes, or using VAX Notes routines in other programs.

Document Structure

This manual is divided into three parts. Part I consists of two chapters, Part II is a dictionary of routines, and Part III consists of three appendixes.

<REFERENCE>(introduction_chap), Introduction, gives general information about the structure of the callable routines.

Chapter 1, VAX Notes Network Protocol, describes the rules that govern the exchange of data between clients and servers.

Part II, VAX Notes Routines Dictionary, provides reference information on the callable routines in alphabetic order.

Appendix A, Item List Format and Contents, provides additional reference information on VAX Notes item codes.

Appendix B, Sample Program, shows and explains a sample program that uses the VAX Notes callable routines. The sample program is presented in the BASIC language.

Appendix C, Messages, lists all error and other messages returned by VAX Notes.

Associated Documents

You should have the *Guide to VAX Notes* (Order number AI-GH98B-TE) available for reference.

Conventions

Convention	Meaning
. . .	A horizontal ellipsis in a figure or example indicates that not all of the statements are shown.
file-spec, . . .	A horizontal ellipsis following a parameter, option, or value in syntax descriptions indicates that additional parameters, options, or values can be entered.
[expression]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional indicators in the syntax of a directory name in a file specification.)
{option|option}	Braces in VAX Notes syntax enclose lists of alternatives from which you must select one item.

Part I

VAX Notes Callable Routines and Network Protocol

1
VAX Notes Network Protocol

The VAX Notes network protocol uses a data syntax based on the DIGITAL Data Interchange Syntax (DDIS).[1]

Data is encoded in TLVs. *TLV* stands for

Type field
Length field
Value field

This chapter describes the TLV encoding in VAX Notes of data packets exchanged between client and server nodes.

TLVs can be nested; that is, the value field of one TLV can contain one or more additional TLVs.

VAX Notes uses unregistered Type codes in its network protocol. The Type codes are the numeric values listed in the file NOTESCODES.TXT, provided with the software kit.

The Type and Length fields of a TLV are bit-encoded for efficiency. The following sections explain the Type, Length, and Value fields.

1.1 Type Field

The Type field of a TLV is bit-encoded in 1, 2, or 3 octets (8-bit bytes).

Bits 6 and 7 of the first or only byte of the Type field specify the class. There are four classes of data types:

0 = TLV$M_UNIVERSAL	Generally useful, application-independent types defined by CCITT X.409

[1] DEC STD 081, DIGITAL Data Interchange Syntax (DDIS), Part 1, as described in A-DS-EL00081-00-3, Rev AX03, 15-Jan-1989.

1 = TLV$M_APPLICATION_WIDE	Specialized types, defined by an application
2 = TLV$M_CONTEXT_SPECIFIC	Application types that are valid only within limited contexts of the application
3 = TLV$M_PRIVATE	Reserved for the private use of Digital

Depending on the context, VAX Notes specifies either TLV$M_PRIVATE or (inappropriately) zero for the class. Thus, bits 6 and 7 of the first byte always contain either 00 (binary) or 11 (binary).

Bit 5 of the first or only byte of the Type field specifies the form. There are two forms of data elements:

0 = TLV$M_PRIMITIVE	An element whose value field is atomic (not itself another data element)
1 = TLV$M_CONSTRUCTOR	An element whose value field is itself a data element or series of data elements

VAX Notes does not specify TLV$M_CONSTRUCTOR, so bit 5 of the first byte is always 0.

The Type field is either 1, 2, or 3 bytes in length, depending on the numeric value of the Type code. The diagrams that follow map the 1-byte, 2-byte, and 3-byte Type fields. In the diagrams,

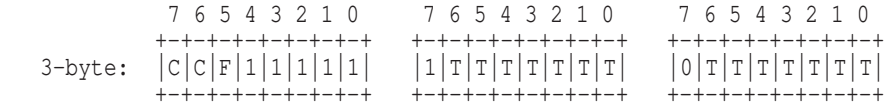
C = Class
F = Form
T = Type
L = Length

```
          7 6 5 4 3 2 1 0
          +-+-+-+-+-+-+-+-+
1-byte:   |C|C|F|T|T|T|T|T|   where TTTTT is less than 11111 (31 decimal)
          +-+-+-+-+-+-+-+-+
```

The one-byte form is used for Types in the range 0 through 30. The number of the Type is in bits 0 through 4. Bit 0 is the least significant bit. The number identifies the routine or item code. (For example, 0001 is the number of NOTES$K_INITIALIZE. This and all other Type codes are listed in the file NOTESCODES.TXT, supplied with your kit.)

```
           7 6 5 4 3 2 1 0     7 6 5 4 3 2 1 0
          +-+-+-+-+-+-+-+-+   +-+-+-+-+-+-+-+-+
2-byte:   |C|C|F|1|1|1|1|1|   |0|T|T|T|T|T|T|T|
          +-+-+-+-+-+-+-+-+   +-+-+-+-+-+-+-+-+
```

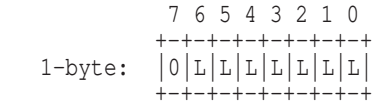
The two-byte form is used for Type codes in the range 31 through 127. Bits 0 through 4 of the first byte are set to 11111. Bit 7 of the second byte is set to 0. The Type code is in bits 0 through 6 of the second byte. Bit 0 is the least significant bit. (For example, 32 decimal is the number of NOTES$K_CLASS_BEGIN).



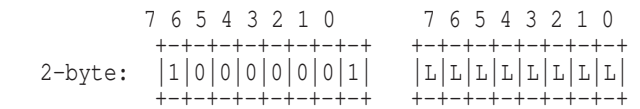
The three-byte form is used for Type codes in the range 128 through 16383. Bit 7 of the second byte is set to 1, and bit 7 of the third byte is set to 0. The Type code is in bits 0 through 6 of the second byte, and bits 0 through 6 of the third byte. Bit 0 of the second byte is the least significant bit, and bit 6 of the third byte is the most significant bit. Hence, to read the binary number, remove bit 7 of the second byte and bit 7 of the third byte; then put the third byte's remaining 7 bits before the second byte's 7 bits, without reordering the bits within each byte. (For example, NOTES$K_INIT_NODENAME is 8194).

1.2 Length Field

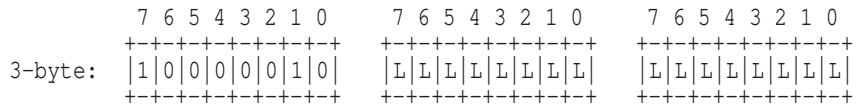
The Length field specifies the length (in 8-bit bytes, or octets) of the Value field of a TLV. The Length field consists of 1, 2, or 3 bytes, in one of the following forms:



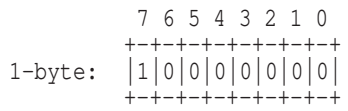
This one-byte form (there is another one-byte form that indicates indefinite length, described below) is used for lengths in the range 0 through 127. Bit 7 is set to 0. The length is specified in bits 0 through 6. Bit 0 is the least significant bit.



The two-byte form is used for lengths in the range 128 through 255. The length is specified in bits 0 through 7 of the second byte. Bit 0 is the least significant bit.



The three-byte form is used for lengths in the range 256 through 65535. The length is specified in bits 0 through 7 of the second byte, and bits 0 through 7 of the third byte. Bit 0 of the second byte is the least significant bit, and bit 7 of the third byte is the most significant bit. (Hence, to read the binary number, put the third byte before the second byte.)



This one-byte form is used to indicate an indefinite length. The Value field that follows is itself TLV-encoded; that is, it is a sequence of TLVs. A TLV of all zeroes within the Value field indicates the end of the Value (that is, both the Type and Length are zero, so this appears as two bytes of zero).

VAX Notes uses this form only in error reporting (see the description of NOTES$K_ERROR_TLV).

1.3 Value Field

For character string data, the Value portion of a TLV is a string of 8-bit octets. Integer data is passed as a 1-octet or 4-octet string with VAX ordering used for the octets within this string. A VMS date/time format is used to pass a date/time data item as an 8-octet string.

1.4 Structure of a Data Packet

The first two bytes of every data packet sent between the VAX Notes client and server contain flags. Currently, only bit 0 of the first byte is used; the other bits are sent as zero and ignored.

Bit 0 of the first byte is the end of message (EOM) bit. When set (1), it indicates that this packet represents the end of the message. When clear (0), it indicates that there are continuation packets that may be requested. Continuation is described in detail below (see the description of NOTES$K_MORE).

Except for continuation packets, the two flag bytes are followed by a sequence of TLVs. The description of these TLVs and their contents comprise the remainder of this chapter.

The protocol design supports only one outstanding request per connection.

In the sequence of TLVs from the server, there may be error-reporting TLVs (see the description of NOTES$K_ERROR_TLV). These indicate an error or informational message that was produced while the server was processing a client request. These may occur anywhere amoung the top-level TLVs; they do not occur in nested TLVs. (A nested TLV is one that occurs as part of the Value field of another TLV.)

To reduce the amount of network traffic, the following two techniques are used in the VAX Notes network protocol.

When a user requests a list (for example, a list of keywords), the callable interface would generally require that multiple calls be made to a GET routine, with NOTES$K_CONTINUE specified on the second and subsequent calls. To reduce the number of packets that are exchanged in handling a user's request, the server assumes that all GET requests are continued, and sends back the information for multiple GET requests in one packet, without requiring the separate requests.

If the information being returned by the server exceeds the size of a single packet, a NOTES$K_MORE request from the client prompts the server to send the next packet in the message. The NOTES$K_USER request illustrates the simplest example of this technique (see the description of NOTES$K_USER_GET).

To further reduce the network traffic, the client can specify "hints" to the server. For example, if the client specifies NOTES$K_NOTE_HINT_GET_TEXT in the NOTES$K_TLV_INPUT TLV of a NOTES$K_NOTE_GET request, then (as usual) the server will send back the items requested in the NOTES$K_TLV_OUTPUT TLV, but it will also send back the text of the note, without the client making individual NOTES$K_NOTE_GET_TEXT requests.

1.4.1 Example: Modifying a User/Membership Record

Most of the ‘‘routine’’ messages are closely related to the corresponding callable interface routines (see Part II. As an example, consider the messages used to modify a user/membership record.

```
USER_MODIFY (from client to server)
```

```
NOTES$K_USER_MODIFY
    NOTES$K_REQUEST_CONTEXT          Context from notefile_begin
    NOTES$K_TLV_INPUT
        NOTES$K_USER_NAME            Username
        NOTES$K_USER_MAIL_ADDR       Mail address
        ...
    NOTES$K_TLV_OUTPUT
        NOTES$K_USER_NAME            Username
        NOTES$K_USER_MODERATE        Longword boolean
        ...
```

The NOTES$K_REQUEST_CONTEXT TLV corresponds to the context parameter that is returned by NOTES$USER_BEGIN and passed to NOTES$USER_MODIFY.

The TLVs within the NOTES$K_TLV_INPUT and NOTES$K_TLV_OUTPUT TLVs correspond directly to the items that can be specified to (or requested from) the NOTES$USER_MODIFY routine. Since their values would be meaningless, the TLVs within the NOTES$K_TLV_OUTPUT TLV are specified with lengths of 0; their presence merely indicates which data items are desired.

The server responds with the following message:

```
USER_MODIFY (from server to client)
    NOTES$K_REQUEST_CONTEXT          Context from notefile_begin
    NOTES$K_TLV_OUTPUT
        NOTES$K_USER_NAME            Username
        NOTES$K_USER_MODERATE        Longword boolean
        ...
    NOTES$K_STATUS                   Longword condition value
```

Note that the server's response is not nested within a NOTES$K_USER_ MODIFY TLV; there is no NOTES$K_TLV_INPUT TLV; and the message includes a NOTES$K_STATUS TLV, which gives the final status of the operation. The NOTES$K_TLV_OUTPUT TLV may be absent (this happens when the NOTES$K_STATUS TLV indicates failure).

None of the BEGIN routines specify the NOTES$K_REQUEST_CONTEXT TLV; this is an output provided by the server's response.

While all the BEGIN responses include the NOTES$K_REQUEST_CONTEXT TLV, most other server responses do not; it is unnecessary because the context is the same as the context in the preceding request (it is assumed that for a given connection the requests are serial).

Responses can also include error or informational messages (see the description of NOTES$K_ERROR_TLV TLV). Some responses may be split into several packets (see the description of NOTES$K_MORE).

1.4.2 Type Numbers of Messages and Routines

The following messages/routines, taken from the complete NOTESCODES.TXT file, are listed with the numbers that identify them in the Type field of TLVs. They fit the description of the messages used to modify a user/membership record, described in Section 1.4.1.

NOTES$K_NOTEFILE_BEGIN = 3
NOTES$K_NOTEFILE_END = 4
NOTES$K_NOTEFILE_GET_INFO = 5
NOTES$K_NOTEFILE_MODIFY = 6
NOTES$K_NOTEFILE_LIST_BEGIN = 7
NOTES$K_NOTEFILE_LIST = 8
NOTES$K_KEYWORD_BEGIN = 9
NOTES$K_KEYWORD_END = 10
NOTES$K_KEYWORD_ADD = 11
NOTES$K_KEYWORD_DELETE = 12
NOTES$K_KEYWORD_GET_NOTE = 14
NOTES$K_KEYWORD_MODIFY = 15
NOTES$K_NOTE_BEGIN = 16
NOTES$K_NOTE_END = 17
NOTES$K_NOTE_ADD = 18
NOTES$K_NOTE_DELETE = 20
NOTES$K_NOTE_GET_KEYWORD = 22
NOTES$K_NOTE_GET_TEXT = 23
NOTES$K_NOTE_MODIFY = 24
NOTES$K_USER_BEGIN = 25
NOTES$K_USER_END = 26
NOTES$K_USER_ADD = 27
NOTES$K_USER_DELETE = 28
NOTES$K_USER_MODIFY = 30
NOTES$K_CLASS_ADD = 31
NOTES$K_CLASS_BEGIN = 32
NOTES$K_CLASS_DELETE = 33
NOTES$K_CLASS_END = 34
NOTES$K_CLASS_GET_ENTRY = 36
NOTES$K_CLASS_MODIFY = 37
NOTES$K_ENTRY_ADD = 38
NOTES$K_ENTRY_BEGIN = 39
NOTES$K_ENTRY_DELETE = 40
NOTES$K_ENTRY_END = 41
NOTES$K_ENTRY_GET_CLASS = 43
NOTES$K_ENTRY_GET_KEYWORD = 44
NOTES$K_ENTRY_MODIFY = 45

NOTES$K_PROFILE_BEGIN = 46
NOTES$K_PROFILE_END = 47
NOTES$K_PROFILE_GET = 48
NOTES$K_PROFILE_MODIFY = 49

1.5 Selected Item Codes and Descriptions

This section describes commonly used item codes in two categories: protocol-specific item codes and "get" item codes.

1.5.1 Protocol-Specific Item Codes

The following protocol-specific item codes are described in this section:

NOTES$K_ERROR_TLV
NOTES$K_INITIALIZE
NOTES$K_NOTE_ADD_TEXT
NOTES$K_MORE
NOTES$K_CANCEL_REQUEST
NOTES$K_REQUEST_CANCELLED

NOTES$K_ERROR_TLV = 131

While processing a client request, the server may produce an error or informational message. To report this, it is put into a NOTES$K_ERROR_ TLV, and is sent to the client as one of the top-level TLVs in the next message (it is not nested within another TLV). The value of the NOTES$K_ERROR_TLV TLV is itself a sequence of TLVs, with the format depending on whether both the client and server specified the NOTES$V_INIT_VMSMSGS capability in their NOTES$K_INITIALIZE messages.

With the NOTES$V_INIT_VMSMSGS capability, the NOTES$K_ERROR_TLV TLV represents a signaled VMS condition vector.

```
NOTES$K_ERROR_TLV
    NOTES$K_STATUS              A longword condition value
    NOTES$K_ERROR_NARGS         Number of following arguments
    NOTES$K_ERROR_ARG           A string argument
```

The NOTES$K_ERROR_ARG TLV may occur any number of times (including 0).

Without the NOTES$V_INIT_VMSMSGS capability:

```
NOTES$K_ERROR_TLV
    NOTES$K_ERROR_TEXT          A string, the error text
```

Note that the indefinite length encoding is used for the NOTES$K_ERROR_ TLV's Length field.

NOTES$K_INITIALIZE = 1

When the client first starts using VAX Notes for a remote connection (by calling NOTES$NOTEFILE_BEGIN), a connection to the server is made, and initialization messages are exchanged.

The initialization message sent by the client contains (in order) the two flag bytes (containing the EOM bit), followed by a TLV of Type NOTES$K_ INITIALIZE. The value of this TLV is itself a sequence of TLVs for the software version, client nodename, client username, and client capabilities. These TLVs are indicated below; note that indentation is used to indicate the nested TLVs.

```
INITIALIZE (from client to server)

    NOTES$K_INITIALIZE
        NOTES$K_INIT_VERSION            Version, a longword integer
        NOTES$K_INIT_NODENAME           Client nodename (a string)
        NOTES$K_INIT_USERNAME           Username (a string)
        NOTES$K_INIT_CAPABILITIES       Capabilities, a longword
```

The server responds with another message. It begins with the flags word, and includes two TLVs, NOTES$K_INITIALIZE and NOTES$K_STATUS. The value of the NOTES$K_INITIALIZE TLV is itself a sequence of TLVs for the software version, server nodename, server capabilities, and the maximum itemcode understood by the server. For VMS, the value of the NOTES$K_ STATUS TLV is a condition value (typically SS$_NORMAL).

```
INITIALIZE (from server to client)

    NOTES$K_INITIALIZE
        NOTES$K_INIT_VERSION            Version, a longword integer
        NOTES$K_INIT_NODENAME           Server nodename (a string)
        NOTES$K_INIT_CAPABILITIES       Capabilities, a longword
        NOTES$K_INIT_MAXITEMCODE        A longword integer
    NOTES$K_STATUS                      A longword condition value
```

Only bit 0 of NOTES$K_INIT_CAPABILITIES is defined; other bits should be clear. If set, bit 0 indicates that the sender (client or server) understands VAX/VMS error messages. This determines the format in which error messages will be sent (see the description of NOTES$K_ERROR_TLV, above). The symbolic name for this capability field within the capabilities longword is NOTES$V_INIT_VMSMSGS.

The first byte of NOTES$K_INIT_VERSION contains the major version number, and the second byte contains the minor version number. For example, if the software version is 2.4, NOTES$K_INIT_VERSION is sent as hex 00020004, which represents version 0002, update 0004. When changes that allow upward compatibility are made to the protocol, the update number is increased; and if significant changes are made, the major version number is increased (and the update reset to 0).

NOTES$K_NOTE_ADD_TEXT = 19

To reduce the number of packets sent when a user writes a note, the client can specify multiple lines of text within a single message.

```
NOTE_ADD_TEXT (from client to server)

    NOTES$K_NOTE_ADD_TEXT
        NOTES$K_TEXT_STRING      \ repeatable
        NOTES$K_TEXT_TYPE        / in any order
        NOTES$K_TEXT_END

NOTE_ADD_TEXT (from server to client)

    NOTES$K_NOTE_ADD_TEXT
        NOTES$K_STATUS
```

NOTES$K_MORE = 2

If a message overflows the buffer size (currently 1024 bytes), then the buffer /packet is sent with the EOM bit clear. The other process can request more (with a NOTES$K_MORE message), and the next packet continues (after the initial two bytes) where the previous one left off.

The MORE message itself is also sent with the EOM bit clear.

```
MORE (from client to server)

    NOTES$K_MORE
        NOTES$K_REQUEST_CONTEXT          Longword context of the request
```

NOTES$K_CANCEL_REQUEST = 135
NOTES$K_REQUEST_CANCELLED = 136

Some server operations may take a long time. To cancel an operation, the client sends a CANCEL_REQUEST message. In response, the server cancels the current operation, and sends back a REQUEST_CANCELLED message. The client should read until it sees the REQUEST_CANCELLED message, since it is possible that the server did not receive the CANCEL_REQUEST message until after it sent its previous response.

```
CANCEL_REQUEST (from client to server)

    NOTES$K_CANCEL_REQUEST                (with Length 0)

CANCEL_REQUEST (from server to client)

    NOTES$K_REQUEST_CANCELLED             (with Length 0)
```

1.5.2 GET Item Codes

The following GET item codes are described in this section:

NOTES$K_USER_GET
NOTES$K_KEYWORD_GET
NOTES$K_CLASS_GET
NOTES$K_ENTRY_GET
NOTES$K_NOTE_GET

NOTES$K_USER_GET = 29

This is requested as described above. The servers response may include multiple NOTES$K_TLV_OUTPUT TLVs, one for each user record found /requested. As usual, NOTES$K_ERROR_TLV TLVs may also appear, and the NOTES$K_TLV_OUTPUT TLV may be absent if the NOTES$K_STATUS TLV indicates failure.

```
USER_GET (from server to client)

    NOTES$K_TLV_OUTPUT                  \
        NOTES$K_USER_NAME                \
        NOTES$K_USER_MODERATE             > This may be repeated
        ...                              /
    NOTES$K_STATUS                      /
```

NOTES$K_KEYWORD_GET = 13

This is requested as described above. The server's response may include multiple NOTES$K_TLV_OUTPUT TLVs, one for each keyword found /requested.

If the NOTES$K_KEYWORD_GET request (sent from the client to the server) includes a NOTES$K_KEYWORD_HINT_GET_NOTE TLV in its NOTES$K_TLV_INPUT TLV, then for each keyword, the server does a NOTES$K_KEYWORD_GET_NOTE and sends that information along in its response. The form of that information depends on whether the NOTES$K_KEYWORD_GET request had NOTES$K_ENTRY_NAME in its NOTES$K_TLV_INPUT TLV.

If NOTES$K_ENTRY_NAME was specified, the server sends a form suited for markers:

```
KEYWORD_GET (from server to client)

     NOTES$K_TLV_OUTPUT                \
         NOTES$K_KEYWORD_NAME           \
         ...                             \
     NOTES$K_STATUS                       \
                                           \
     NOTES$K_TLV_OUTPUT     \ one for       > This may be repeated
         NOTES$K_NOTE_ID     \ each         /
         NOTES$K_ENTRY_NAME   \ note       /
         NOTES$K_KEYWORD_NAME /           /
         NOTES$K_NOTE_UID    /           /
     NOTES$K_STATUS         /           /
```

If NOTES$K_ENTRY_NAME was not specified:

```
KEYWORD_GET (from server to client)

     NOTES$K_TLV_OUTPUT                  \
         NOTES$K_KEYWORD_NAME             \
         ...                               \
     NOTES$K_STATUS                         \
                                             > This may be repeated
     NOTES$K_NOTE_UID       \ one for       /
     NOTES$K_NOTE_ID        / each note    /
                                          /
     NOTES$K_STATUS - end of all the     /
                      hooji-woojis      /
```

The first NOTES$K_STATUS shown above is for the NOTES$K_KEYWORD_GET operation, and the second is for the final (unsuccessful) NOTES$K_KEYWORD_GET_NOTE done for each keyword.

The NOTES$K_NOTE_UID may be absent, since early versions of VAX Notes (prior to protocol 2/3) did not include it.

NOTES$K_CLASS_GET = 35

This is requested as described above. The servers response may include multiple NOTES$K_TLV_OUTPUT TLVs, one for each class record found /requested.

If the NOTES$K_CLASS_GET request includes a NOTES$K_CLASS_HINT_GET_ENTRY TLV in its NOTES$K_TLV_INPUT TLV, then for each class, the server does a NOTES$K_CLASS_GET_ENTRY and sends the results in its response.

```
CLASS_GET (from server to client)

    NOTES$K_TLV_OUTPUT                    \
        NOTES$K_CLASS_NAME                 \
        ...                                 \
    NOTES$K_STATUS                           \ This may be repeated
                                             /
    NOTES$K_ENTRY_NAME  > repeated          /
                                           /
    NOTES$K_STATUS                        /
```

The first NOTES$K_STATUS shown above is for the NOTES$K_CLASS_GET operation, and the second is for the final (unsuccessful) NOTES$K_CLASS_ GET_ENTRY done for each class.

NOTES$K_ENTRY_GET = 42

In a NOTES$K_ENTRY_GET request a client may specify one of two hints, NOTES$K_ENTRY_HINT_GET_CLASS or NOTES$K_ENTRY_HINT_GET_ KEYWORD, but not both.

If NOTES$K_ENTRY_HINT_GET_CLASS is specified, then for each entry, the server does a NOTES$K_ENTRY_GET_CLASS and sends the results in its response.

```
ENTRY_GET (from server to client)

    NOTES$K_TLV_OUTPUT                    \
        NOTES$K_ENTRY_NAME                 \
        ...                                 \
    NOTES$K_STATUS                           \ This may be repeated
                                             /
    NOTES$K_CLASS_NAME   > repeated         /
                                           /
    NOTES$K_STATUS                        /
```

If NOTES$K_ENTRY_HINT_GET_KEYWORD is specified, then for each entry, the server does a NOTES$K_ENTRY_GET_KEYWORD and sends the results in its response.

```
ENTRY_GET (from server to client)

    NOTES$K_TLV_OUTPUT                    \
        NOTES$K_ENTRY_NAME                 \
        ...                                 \
    NOTES$K_STATUS                           \ This may be repeated
                                             /
    NOTES$K_KEYWORD_NAME   > repeated       /
                                           /
    NOTES$K_STATUS                        /
```

NOTES$K_NOTE_GET = 21

This is requested as described above. The server's response may include multiple NOTES$K_TLV_OUTPUT TLVs, one for each note found/selected.

If the NOTES$K_NOTE_GET request includes a NOTES$K_NOTE_HINT_GET_KEYWORD TLV in its NOTES$K_TLV_INPUT TLV, then for each note, the server does NOTES$K_NOTE_GET_KEYWORD operations and sends the resulting keywords in its response. If the request includes a NOTES$K_NOTE_HINT_GET_TEXT TLV, then for each note, the server does NOTES$K_NOTE_GET_TEXT operations, and sends the results in its response. The two hints cannot be specified together.

If NOTES$K_NOTE_GET_KEYWORD is specified:

```
NOTE_GET (from server to client)

    NOTES$K_TLV_OUTPUT                    \
        NOTES$K_NOTE_UID                   \
        ...                                 \
    NOTES$K_STATUS                           \ This may be repeated
                                             /
    NOTES$K_KEYWORD_NAME   > repeated       /
                                           /
    NOTES$K_STATUS                        /
```

If NOTES$K_NOTE_GET_TEXT is specified:

```
NOTE_GET (from server to client)

    NOTES$K_TLV_OUTPUT                  \
        NOTES$K_NOTE_UID                 \
        ...                               \
    NOTES$K_STATUS                         \
                                            > This may be repeated
    NOTES$K_TEXT_TYPE      \ repeated      /
    NOTES$K_TEXT_STRING    /              /
                                         /
    NOTES$K_STATUS                      /
```

The NOTES$K_TEXT_TYPE may be absent, since early versions of VAX Notes (prior to protocol 2/3) did not include it.

The first NOTES$K_STATUS shown above (in each diagram) is for the NOTES$K_NOTE_GET operation, and the second is for the final (unsuccessful) NOTES$K_NOTE_GET_KEYWORD or NOTES$K_NOTE_GET_TEXT operation done for each note.

If NOTES$K_NOTE_SEARCH_STRING is specified with NOTES$K_NOTE_GET_TEXT, then the server empties/sends its buffer at the end of each note, rather than waiting until the buffer fills. This is consistent with the protocol, and gives better user-perceived performance, as the results of a long-lasting SEARCH command are sent to the client as soon as they are available, rather than being further delayed until the buffer is full or all the notes have been searched.

Part II

VAX Notes Routines Dictionary

1 General Information

Part II is a dictionary of the VAX Notes routines. The routines are arranged alphabetically so that you can find them quickly without consulting the table of contents or the index. Descriptions of their format, arguments, and return codes are included.

This section contains the information that is common to all the routines: the attributes of their arguments in VMS and in ULTRIX, and the item codes permitted in the output item lists of the routines in each functional category. (The *input* item lists are unique for each routine, and thus are in the individual routine descriptions.)

1.1 Attributes of Routine Arguments in VMS and ULTRIX

Table 1 and Table 2 show, respectively, the attributes of the arguments to the Notes routines in VMS and in ULTRIX.

Note

The argument to the NOTES$NOTES routine has different attributes, which are described in the NOTES$NOTES section in this routines dictionary.

Table 1 Attributes in VMS of Arguments to Notes Routines

Argument	VMS Usage	Type	Access	Mechanism
Context	context	longword (unsigned)	modify	by refer-ence
Input item list	item_list_3	longword (unsigned)	read only	by refer-ence
Output item list	item_list_3	longword (unsigned)	read only	by refer-ence

Table 2 Attributes in ULTRIX of Arguments to Notes Routines

Argument	ULTRIX Usage	Type	Access	Mechanism
Context	TBS...			
Input item list				
Output item list				

1.2 Categories of Routines

There are eight functional categories of Notes routines. The category of each routine appears in its name after the NOTES$ prefix (for example, NOTES$CLASS_ADD is a Class routine). The eight categories are as follows:

Class
Entry
Keyword
Note
Notefile
Notes[1]
Profile
User

2 General Rules for Routine Formats

The format (syntax) of each routine appears under the routine. Certain general rules apply throughout, however, as follows:

1. The name of a routine (always beginning with NOTES$) is followed by a space and then the arguments enclosed in parentheses and separated by commas.
2. The context argument is required for all routines, and comes first.
3. The input item list follows the context, and is required for all the routines except the END routines, for which it is optional.
4. The output item list is last, and is optional for all the routines.
5. When you choose not to specify an optional input or output item list, you must pass a zero in its place.

[1] The Notes category consists of a single routine, NOTES$NOTES, which invokes Notes interactively. The NOTES$NOTES routine is discussed in the <REFERENCE>(VAX_NOTES_GT), as well as in Part II.

Hence, the following format is applicable to all the routines except the END routines:

NOTES$name (context,input-item-list[,output-item-list])

Following is the format for the END routines:

NOTES$name_END (context[,input-item-list[,output-item-list]])

Hence, the following format is applicable to all the routines except the END routines:

NOTES$name (context,input-item-list,{output-item-list|0})

Following is the format for the END routines:

NOTES$name_END (context,{input-item-list|0},{output-item-list|0})

2.1 Valid Item Codes for Output Item Lists

The item codes permitted in the output item list are the same (with a few exceptions, noted) for each category of routines. The following sections list the permitted output item codes. Note that all output item codes are optional.

Class

NOTES$K_CLASS_NAME
NOTES$K_CLASS_USER_AREA
NOTES$K_ENTRY_NAME (for NOTES$CLASS_GET_ENTRY calls)

Entry

NOTES$K_CLASS_NAME (for NOTES$ENTRY_GET_CLASS calls)
NOTES$K_ENTRY_LAST_STATUS
NOTES$K_ENTRY_NAME
NOTES$K_ENTRY_OBJECT_NAME
NOTES$K_ENTRY_UNSEEN_EST
NOTES$K_ENTRY_USER_AREA
NOTES$K_KEYWORD_NAME (for NOTES$ENTRY_GET_KEYWORD calls)
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_ENTRYTOTAL
NOTES$K_NOTEFILE_FILE_NAME
NOTES$K_NOTEFILE_HIGH_UID
NOTES$K_NOTEFILE_LASTREV
NOTES$K_NOTEFILE_MODERATOR

NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_NUMNOTES
NOTES$K_NOTEFILE_RELATED_NAME
NOTES$K_NOTEFILE_TITLE
NOTES$K_SEEN_MAP

Keyword

NOTES$K_ENTRY_NAME (for NOTES$KEYWORD_GET_NOTE calls)
NOTES$K_KEYWORD_NAME
NOTES$K_KEYWORD_USER_AREA
NOTES$K_NOTE_ID (for NOTES$KEYWORD_GET_NOTE calls)
NOTES$K_NOTE_UID (for NOTES$KEYWORD_GET_NOTE calls)

Note

NOTES$K_KEYWORD_NAME (for NOTES$NOTE_GET_KEYWORD calls)
NOTES$K_NOTE_AUTHOR
NOTES$K_NOTE_BLINK_ID
NOTES$K_NOTE_BLINK_TITLE
NOTES$K_NOTE_BLINK_UID
NOTES$K_NOTE_CAN_REPLY
NOTES$K_NOTE_CREATE_TIME
NOTES$K_NOTE_HIDDEN
NOTES$K_NOTE_ID
NOTES$K_NOTE_NUMRECORDS
NOTES$K_NOTE_NUMRESPONSES
NOTES$K_NOTE_PEN_NAME
NOTES$K_NOTE_TITLE
NOTES$K_NOTE_TYPE
NOTES$K_NOTE_UID
NOTES$K_NOTE_USER_AREA
NOTES$K_NOTE_WRITELOCK
NOTES$K_NOTEFILE_ENTRYTOTAL
NOTES$K_NOTEFILE_FILE_NAME
NOTES$K_NOTEFILE_HIGH_UID
NOTES$K_NOTEFILE_LASTREV
NOTES$K_NOTEFILE_NUMNOTES
NOTES$K_SEEN_MAP
NOTES$K_TEXT_TYPE (for NOTES$NOTE_GET_TEXT and NOTES$NOTE_ADD_TEXT calls)
NOTES$K_TEXT_STRING (for NOTES$NOTE_GET_TEXT and NOTES$NOTE_ADD_TEXT calls)

Notefile

NOTES$K_ENTRY_OBJECT_SPEC
NOTES$K_NOTEFILE_CREATE_TIME
NOTES$K_NOTEFILE_ENTRYTOTAL
NOTES$K_NOTEFILE_FORMAT
NOTES$K_NOTEFILE_HIGH_UID
NOTES$K_NOTEFILE_LASTREV
NOTES$K_NOTEFILE_MODERATOR
NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_NUMNOTES
NOTES$K_NOTEFILE_REPLY_ONLY
NOTES$K_NOTEFILE_RESTRICTED
NOTES$K_NOTEFILE_RESULT_SPEC
NOTES$K_NOTEFILE_TITLE
NOTES$K_NOTEFILE_USER_AREA
NOTES$K_NOTEFILE_WRITELOCK
NOTES$K_SEEN_MAP
NOTES$K_SERVER_MAX_ITEM
NOTES$K_USER_CREATE_KEYWORD
NOTES$K_USER_MODERATE

Profile

NOTES$K_PROFILE_AUTO_DIR
NOTES$K_PROFILE_AUTO_UNSEEN
NOTES$K_PROFILE_CLASS_NAME
NOTES$K_PROFILE_EDITOR
NOTES$K_PROFILE_EDITOR_SPAWN
NOTES$K_PROFILE_PEN_NAME
NOTES$K_PROFILE_PRINT

User

NOTES$K_USER_ACCESS_LIST
NOTES$K_USER_CREATE_KEYWORD
NOTES$K_USER_MAIL_ADDR
NOTES$K_USER_MODERATE
NOTES$K_USER_NAME
NOTES$K_USER_NODENAME
NOTES$K_USER_WRITE_BYPASS

NOTES$CLASS_ADD

Creates a class record if the entry name is missing. If the entry name is specified, NOTES$CLASS_ADD adds that entry to the class.

Format

```
NOTES$CLASS_ADD  (context,input-item-list
                        [,output-item-list])
```

Arguments

context
The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_CLASS_NAME

Optional:

NOTES$K_CLASS_USER_AREA
NOTES$K_ENTRY_NAME
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_ADD routine creates a class record if the entry name is missing. If the entry name is specified, NOTES$CLASS_ADD adds that entry to the class.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_BEGIN

Creates and initializes the class context for later calls to NOTES$CLASS_xxx routines.

Format

```
NOTES$CLASS_BEGIN   (context,input-item-list
                            [,output-item-list])
```

Arguments

context
The address of a longword that will receive class context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_BEGIN routine creates and initializes the class context for later calls to NOTES$CLASS_xxx routines.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_DELETE

Deletes the class record if the entry name is missing. If the entry name is specified, NOTES$CLASS_DELETE deletes that entry from the class.

Format

```
NOTES$CLASS_DELETE  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_CLASS_NAME

Optional:

NOTES$K_DELIFO
NOTES$K_ENTRY_NAME
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_DELETE routine deletes the class record if the entry name is missing. If the entry name is specified, NOTES$CLASS_DELETE deletes that entry from the class.

NOTES$CLASS_DELETE

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_END

Deletes or deallocates the class context.

Format

```
NOTES$CLASS_END   (context[,input-item-list]
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_END routine deletes or deallocates the class context. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_GET

Gets information about a class or, if NOTES$K_CONTINUE was specified, the next class.

Format

```
NOTES$CLASS_GET  (context,input-item-list
                        [,output-item-list])
```

Arguments

context

The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_CLASS_NAME

Optional:

NOTES$K_CLASS_HINT_GET_ENTRY
NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_GET routine gets information about a class or, if NOTES$K_CONTINUE was specified, the next class.

Return Code	Description	Status
NOTES$_?		

NOTES$CLASS_GET

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_GET_ENTRY

Gets information about entries belonging to the class specified by the most recent NOTES$CLASS_GET call for this context.

Format

```
NOTES$CLASS_GET_ENTRY   (context,input-item-list
                                 [,output-item-list])
```

Arguments

context
The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_GET_ENTRY routine gets information about entries belonging to the class specified by the most recent NOTES$CLASS_GET call for this context.

Return Code	Description	Status
NOTES$_?		

NOTES$CLASS_GET_ENTRY

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$CLASS_MODIFY

Modifies information stored for a class.

Format

```
NOTES$CLASS_MODIFY  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the class context returned by NOTES$CLASS_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_CLASS_NAME

Optional:

NOTES$K_CLASS_NEW_NAME
NOTES$K_CLASS_USER_AREA
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$CLASS_MODIFY routine modifies the information stored for a class.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$CLASS_MODIFY

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_ADD

Adds an entry to the specified class.

Format

```
NOTES$ENTRY_ADD  (context,input-item-list
                          [,output-item-list])
```

Arguments

context

The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_ENTRY_NAME
NOTES$K_NOTEFILE_FILE_NAME

Optional:

NOTES$K_CLASS_NAME
NOTES$K_ENTRY_LAST_STATUS
NOTES$K_ENTRY_OBJECT_NAME
NOTES$K_ENTRY_USER_AREA
NOTES$K_NOSIGNAL
NOTES$K_NOTE_BEFORE_TIME
NOTES$K_NOTE_SINCE_TIME
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_ENTRYTOTAL
NOTES$K_NOTEFILE_HIGH_UID
NOTES$K_NOTEFILE_LASTREV
NOTES$K_NOTEFILE_MODERATOR
NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_NUMNOTES
NOTES$K_NOTEFILE_RELATED_NAME
NOTES$K_NOTEFILE_TITLE
NOTES$K_SEEN_MAP

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_ADD routine adds an entry to the specified class.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_BEGIN

Creates and initializes the entry context for later calls to NOTES$ENTRY_xxx routines.

Format

```
NOTES$ENTRY_BEGIN   (context,input-item-list
                            [,output-item-list])
```

Arguments

context
The address of a longword that will receive entry context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_BEGIN routine creates and initializes the entry context for later calls to NOTES$ENTRY_xxx routines.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_DELETE

Deletes the entry record.

Format

```
NOTES$ENTRY_DELETE  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_ENTRY_NAME

Optional:

NOTES$K_CLASS_NAME
NOTES$K_DELIFO
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_DELETE routine deletes the entry record.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_DELETE

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_END

Deletes or deallocates the entry context.

Format

```
NOTES$ENTRY_END   (context[,input-item-list]
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_END routine deletes or deallocates the entry context. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_GET

Gets information from the entry record.

Format

```
NOTES$ENTRY_GET  (context,input-item-list
                          [,output-item-list])
```

Arguments

context

The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_ENTRY_NAME

Optional:

NOTES$K_CLASS_NAME
NOTES$K_CONTINUE
NOTES$K_ENTRY_HINT_GET_CLASS
NOTES$K_ENTRY_HINT_GET_KEYWORD
NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_GET routine gets information from the entry record.

Return Code	Description	Status
NOTES$_?		

NOTES$ENTRY_GET

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_GET_CLASS

Gets information about the classes to which an entry belongs (the entry specified by the most recent NOTES$ENTRY_GET call for the context).

Format

```
NOTES$ENTRY_GET_CLASS   (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_GET_CLASS routine gets information about the classes to which an entry belongs (the entry specified by the most recent NOTES$ENTRY_GET call for the context).

Return Code	Description	Status
NOTES$_?		

NOTES$ENTRY_GET_CLASS

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_GET_KEYWORD

Gets information about keywords or markers for an entry.

Format

```
NOTES$ENTRY_GET_KEYWORD  (context,input-item-list
                                    [,output-item-list])
```

Arguments

context
The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_GET_KEYWORD routine gets information about keywords or markers for an entry.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_GET_KEYWORD

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_MODIFY

Modifies the information stored for an entry.

Format

```
NOTES$ENTRY_MODIFY  (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_ENTRY_NAME

Optional:

NOTES$K_ENTRY_LAST_STATUS
NOTES$K_ENTRY_NEW_NAME
NOTES$K_ENTRY_OBJECT_NAME
NOTES$K_ENTRY_USER_AREA
NOTES$K_NOSIGNAL
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_ENTRYTOTAL
NOTES$K_NOTEFILE_FILE_NAME
NOTES$K_NOTEFILE_HIGH_UID
NOTES$K_NOTEFILE_LASTREV
NOTES$K_NOTEFILE_MODERATOR
NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_NUMNOTES
NOTES$K_NOTEFILE_RELATED_NAME
NOTES$K_NOTEFILE_TITLE
NOTES$K_SEEN MAP

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_MODIFY routine modifies the information stored for an entry.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$ENTRY_UPDATE

Updates statistics for an entry (or entries, if wildcards are used).

Format

```
NOTES$ENTRY_UPDATE  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the entry context returned by NOTES$ENTRY_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_ENTRY_NAME

Optional:

NOTES$K_CLASS_NAME
NOTES$K_CONTINUE
NOTES$K_NOSIGNAL
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_FILE_NAME
NOTES$K_NOTEFILE_RELATED_NAME

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$ENTRY_UPDATE routine updates statistics for an entry (or entries, if wildcards are used). The unseen count is only an estimate of the notes unseen.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_ADD

Creates a keyword record if both the entry name and the note-ID are missing; creates a keyword reference if either the entry name or the note-ID is specified.

Format

```
NOTES$KEYWORD_ADD   (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_KEYWORD_NAME

Optional:

NOTES$K_ENTRY_NAME
NOTES$K_KEYWORD_USER_AREA
NOTES$K_NOSIGNAL
NOTES$K_NOTE_ID
NOTES$K_NOTE_UID
NOTES$K_UNIENTRY

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_ADD routine creates a keyword record if both the entry name and the note-ID are missing; it creates a keyword reference if either the entry name or the note-ID is specified. If the entry name is not specified, it also adds the reference to the note (if specified).

Note that this routine and all other keyword routines are used for markers as well as keywords.

NOTES$KEYWORD_ADD

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_BEGIN

Creates and initializes the keyword context for later calls to NOTES$KEYWORD_xxx routines.

Format

```
NOTES$KEYWORD_BEGIN  (context,input-item-list
                              [,output-item-list])
```

Arguments

context
The address of a longword that will receive keyword context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_BEGIN routine creates and initializes the keyword context for later calls to NOTES$KEYWORD_xxx routines.

Note that this routine and all other keyword routines are used for markers as well as keywords.

Return Code	Description	Status
NOTES$_?		

NOTES$KEYWORD_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_DELETE

Deletes a keyword record if neither the entry name nor the note-ID is specified. If either the entry name or the note-ID is specified, this routine deletes the keyword reference; and if the entry name is not specified and the note-ID is specified, it also deletes the keyword reference from the note.

Format

```
NOTES$KEYWORD_DELETE   (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_KEYWORD_NAME

Optional:

NOTES$K_ENTRY_NAME
NOTES$K_NOSIGNAL
NOTES$K_NOTE_ID
NOTES$K_NOTE_UID

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_DELETE routine deletes a keyword record if neither the entry name nor the note-ID is specified. If either the entry name or the note-ID is specified, this routine deletes the keyword reference; and if the entry name is not specified and the note-ID is specified, it also deletes the keyword reference from the note.

NOTES$KEYWORD_DELETE

Note that this routine and all other keyword routines are used for markers as well as keywords.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_END

Deletes or deallocates the keyword context.

Format

```
NOTES$KEYWORD_END   (context[,input-item-list]
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_END routine deletes or deallocates the keyword context. The context longword is zeroed.

Note that this routine and all other keyword routines are used for markers as well as keywords.

Return Code	Description	Status
NOTES$_?		

NOTES$KEYWORD_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_GET

Gets information about keywords or markers.

Format

```
NOTES$KEYWORD_GET  (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_KEYWORD_NAME

Optional:

NOTES$K_CONTINUE
NOTES$K_ENTRY_NAME
NOTES$K_KEYWORD_HINT_GET_NOTE (necessary if NOTES$KEYWORD_GET_NOTE is to follow NOTES$KEYWORD_GET)
NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_GET routine gets information about keywords or markers.

Note that this routine and all other keyword routines are used for markers as well as keywords.

NOTES$KEYWORD_GET

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_GET_NOTE

Gets information about notes referenced by a keyword (the keyword specified by the most recent NOTES$KEYWORD_GET call for this context) or by a marker.

Format

```
NOTES$KEYWORD_GET_NOTE  (context,input-item-list
                                    [,output-item-list])
```

Arguments

context
The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_GET_NOTE routine gets information about notes referenced by a keyword (the keyword specified by the most recent NOTES$KEYWORD_GET call for this context) or by a marker.

Note that this routine and all other keyword routines are used for markers as well as keywords.

NOTES$KEYWORD_GET_NOTE

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$KEYWORD_MODIFY

Changes a keyword name.

Format

```
NOTES$KEYWORD_MODIFY  (context,input-item-list
                                    [,output-item-list])
```

Arguments

context
The address of a longword that contains the keyword context returned by NOTES$KEYWORD_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_KEYWORD_NAME

Optional:

NOTES$K_KEYWORD_NEW_NAME
NOTES$K_KEYWORD_USER_AREA
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$KEYWORD_MODIFY routine changes a keyword name. References are created and deleted by calls to the NOTES$KEYWORD_ADD and NOTES$KEYWORD_DELETE routines.

If the keyword_new_name does not exist, NOTES$KEYWORD_MODIFY udates the keyword record. Otherwise, the routine effectively uses NOTES$KEYWORD_GET, NOTES$KEYWORD_ADD, and NOTES$KEYWORD_DELETE to copy all references from the keyword_name to the keyword_new_name, and then deletes the keyword_name record.

NOTES$KEYWORD_MODIFY

Note that this routine and all other keyword routines are used for markers as well as keywords.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_ADD

Adds a new note to a notefile.

Format

```
NOTES$NOTE_ADD  (context,input-item-list
                        [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_NOTE_BLINK_ID
NOTES$K_NOTE_BLINK_UID
NOTES$K_NOTE_HIDDEN
NOTES$K_NOTE_MARK_SEEN
NOTES$K_NOTE_PEN_NAME
NOTES$K_NOTE_TITLE
NOTES$K_NOTE_USER_AREA
NOTES$K_NOTE_WRITELOCK
NOTES$K_NOTEFILE_FILE_NAME

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_ADD routine adds a new note (either topic or reply) to a notefile. Calls to the NOTES$NOTE_ADD_TEXT routine can then be used to write the text of the note.

The BLINK (back-link) item codes, if present, indicate the note to which the reply is being written. If the BLINK item codes are not specified, the note is a topic rather than a reply.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_ADD_TEXT

Adds a line of text to the note being written. After all the text has been added by calls to this routine, it should be called again, with the NOTES$K_TEXT_END item code specified.

Format

```
NOTES$NOTE_ADD_TEXT   (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL
NOTES$K_TEXT_END
NOTES$K_TEXT_STRING
NOTES$K_TEXT_TYPE

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_ADD_TEXT routine adds a line of text to the note being written. After all the text has been added by calls to this routine, it should be called again, with the NOTES$K_TEXT_END item code specified.

NOTES$NOTE_ADD_TEXT

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_BEGIN

Initiates note-level access to a currently open notefile.

Format

```
NOTES$NOTE_BEGIN  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that will receive note context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_BEGIN routine initiates note-level access to a currently open notefile. It creates and initializes the note context for later calls to NOTES$NOTE_xxx routines.

The NOTES_NOTE_BEGIN routine must be called before any of the other NOTES$NOTE_xxx routines and after the NOTES$NOTEFILE_BEGIN routine has been called to open the notefile.

It is possible to have more than one NOTE_BEGIN/END routine active in the same NOTEFILE_BEGIN/END block.

Note that this routine and all other keyword routines are used for markers as well as keywords.

NOTES$NOTE_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_DELETE

Deletes the specified topic or reply.

Format

```
NOTES$NOTE_DELETE  (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_NOTE_ID
NOTES$K_NOTE_UID

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_DELETE routine deletes the specified topic or reply.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$NOTE_DELETE

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_END

Deletes or deallocates the note context.

Format

```
NOTES$NOTE_END  (context[,input-item-list]
                        [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_END routine deletes or deallocates the note context. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_GET

Retrieves information about a note or sequence of notes.

Format

```
NOTES$NOTE_GET  (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_KEYWORD_NAME
NOTES$K_NOSIGNAL
NOTES$K_NOTE_ALL_RESPONSES
NOTES$K_NOTE_AUTHOR
NOTES$K_NOTE_BACK_NOTE
NOTES$K_NOTE_BACK_RESPONSE
NOTES$K_NOTE_BEFORE_TIME
NOTES$K_NOTE_HINT_GET_KEYWORD
NOTES$K_NOTE_HINT_GET_TEXT
NOTES$K_NOTE_ID
NOTES$K_NOTE_MARK_SEEN
NOTES$K_NOTE_NEXT_NOTE
NOTES$K_NOTE_NEXT_RESPONSE
NOTES$K_NOTE_NEXT_UNSEEN
NOTES$K_NOTE_SEARCH_STRING
NOTES$K_NOTE_SINCE_TIME
NOTES$K_NOTE_TITLE
NOTES$K_NOTE_UID
NOTES$K_NOTE_UNSEEN

NOTES$K_NOTE_USER_AREA

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_GET routine retrieves information about a note or sequence of notes. It must be called before any calls to the NOTES$NOTE_GET_KEYWORD or the NOTES$NOTE_GET_TEXT routines.

The item codes NOTES$K_NOTE_BACK_NOTE, NOTES$K_NOTE_NEXT_NOTE, NOTES$K_NOTE_BACK_RESPONSE, and NOTES$K_NOTE_NEXT_RESPONSE conflict with each other and with all other item codes.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_GET_KEYWORD

Gets the keywords associated with the current note (the note specified by the last call to the NOTES$NOTE_GET routine). Each call returns one keyword.

Format

```
NOTES$NOTE_GET_KEYWORD  (context,input-item-list
                                 [,output-item-list])
```

Arguments

context
The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_GET_KEYWORD routine gets the keywords associated with the current note (the note specified by the last call to the NOTES$NOTE_ GET routine). Each call returns one keyword.

Return Code	Description	Status
NOTES$_?		

NOTES$NOTE_GET_KEYWORD

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_GET_TEXT

Gets the text records associated with the current note (the note specified by the last call to the NOTES$NOTE_GET routine). Each call returns one text record.

Format

```
NOTES$NOTE_GET_TEXT  (context,input-item-list
                            [,output-item-list])
```

Arguments

context
The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_GET_TEXT routine gets the text records associated with the current note (the note specified by the last call to the NOTES$NOTE_GET routine). Each call returns one text record.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTE_MODIFY

Modifies the attributes of a note.

Format

```
NOTES$NOTE_MODIFY  (context,input-item-list
                          [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$NOTE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_NOTE_HIDDEN
NOTES$K_NOTE_ID
NOTES$K_NOTE_NEW_ID
NOTES$K_NOTE_PEN_NAME
NOTES$K_NOTE_TITLE
NOTES$K_NOTE_UID
NOTES$K_NOTE_UNSEEN
NOTES$K_NOTE_USER_AREA
NOTES$K_NOTE_WRITELOCK
NOTES$K_NOTEFILE_FILE_NAME

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTE_MODIFY routine modifies the attributes of a note.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_BEGIN

Initiates access to a notefile (conference or Notebook).

Format

```
NOTES$NOTEFILE_BEGIN   (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that will receive notefile context information (as described in the Description section).

input-item-list

An item list selected from the following:

Required:

NOTES$K_NOTEFILE_FILE_NAME

Optional:

NOTES$K_ENTRY_OBJECT_NAME
NOTES$K_NOSIGNAL
NOTES$K_NOTEFILE_CREATE
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_MODERATE
NOTES$K_NOTEFILE_MODERATOR
NOTES$K_NOTEFILE_RELATED_NAME
NOTES$K_NOTEFILE_REPLY_ONLY
NOTES$K_SEEN_MAP
NOTES$K_USER_CREATE_KEYWORD

The following item codes are allowed as input items when NOTES$K_ NOTEFILE_CREATE is passed to the NOTES$NOTEFILE_BEGIN routine.

NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_RESTRICTED
NOTES$K_NOTEFILE_TITLE
NOTES$K_NOTEFILE_USER_AREA
NOTES$K_NOTEFILE_WRITELOCK
NOTES$K_USER_MODERATE

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTEFILE_END routine deletes or deallocates a notefile context. This routine is used to terminate access to a notefile. All dynamic memory allocated by the NOTES$ routines when accessing the notefile (including CLASS, ENTRY, KEYWORD, PROFILE, and USER access) is deallocated. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_GET_INFO

Obtains information about a notefile.

Format

```
NOTES$NOTEFILE_GET_INFO  (context,input-item-list
                                    [,output-item-list])
```

Arguments

context

The address of a longword that contains the notefile context returned by NOTES$NOTEFILE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTEFILE_GET_INFO routine obtains information about a notefile.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_GET_INFO

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_LIST

Collects information on a selected group of notefiles.

Format

```
NOTES$NOTEFILE_LIST   (context,input-item-list
                                    [,output-item-list])
```

Arguments

context

The address of a longword that that contains the notefile context returned by NOTES$NOTEFILE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTEFILE_LIST routine collects information on a selected group of notefiles. The NOTES$NOTEFILE_LIST_BEGIN routine must be called before the NOTES$NOTEFILE_LIST routine. You specify the filenames in the NOTES$NOTEFILE_LIST_BEGIN routine, and collect the information on successive calls to NOTES$NOTEFILE_LIST until the status NOTES$_NMF is returned.

Any output item codes specified in the call to the NOTES$NOTEFILE_LIST routine must also be specified in the call to the NOTES$NOTEFILE_LIST_ BEGIN routine for correct operation on remote conference directories.

NOTES$NOTEFILE_LIST

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_LIST_BEGIN

Initiates the process of collecting notefile names for a directory listing of conferences.

Format

```
NOTES$NOTEFILE_LIST_BEGIN   (context,input-item-list
                                        [,output-item-list])
```

Arguments

context
The address of a longword that contains the notefile context returned by NOTES$NOTEFILE_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_FILE_NAME

Optional:

NOTES$K_NOSIGNAL
NOTES$K_NOTEFILE_DEFAULT_NAME
NOTES$K_NOTEFILE_RELATED_NAME

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTEFILE_LIST_BEGIN routine initiates the process of collecting notefile names, for a directory listing of conferences. You call the NOTES$NOTEFILE_LIST_BEGIN routine with the file specification, default file specification, and related file specification. It performs the initialization. After a successful call to NOTES$NOTEFILE_LIST_BEGIN you call NOTES$NOTEFILE_LIST until the status NOTES$_NMF is returned.

NOTES$NOTEFILE_LIST_BEGIN

For server performance considerations, in the output item list for the NOTES$NOTEFILE_LIST_BEGIN routine you must specify all the items that you specify in the call to NOTES$NOTEFILE_LIST. However, the items are not returned on the call to the NOTES$NOTEFILE_LIST_BEGIN routine.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTEFILE_MODIFY

Updates or changes notefile-level information, such as the title, notice of the day, and moderator.

Format

```
NOTES$NOTEFILE_MODIFY   (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the notefile context returned by NOTES$NOTEFILE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_NOTE_BEFORE_TIME
NOTES$K_NOTE_MARK_SEEN
NOTES$K_NOTE_SINCE_TIME
NOTES$K_NOTEFILE_MODERATE
NOTES$K_NOTEFILE_MODERATOR
NOTES$K_NOTEFILE_NOTICE
NOTES$K_NOTEFILE_REPLY_ONLY
NOTES$K_NOTEFILE_RESTRICTED
NOTES$K_NOTEFILE_TITLE
NOTES$K_NOTEFILE_USER_AREA
NOTES$K_NOTEFILE_WRITELOCK
NOTES$K_SEEN_MAP
NOTES$K_USER_CREATE_KEYWORD

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$NOTEFILE_MODIFY routine updates or changes notefile-level information, such as the title, notice of the day, and moderator.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$NOTES

Invokes the interactive VAX Notes product, with the appropriate user interface based on the input device, and processes commands from SYS$INPUT until it returns to the calling program. You should generally call NOTES$NOTES as a function, using the format below, rather than as a procedure, because the function returns a condition value, without which there can be undiscovered errors.

Format

condition_value = NOTES$NOTES ([arg1])

Arguments

arg1
The address of a VMS string descriptor of the VAX Notes DCL command line with which to start up VAX Notes. This argument must have the following attributes:

Attribute	Value	Description
VMS Usage:	char_string	VMS data structure; string of from 0 to 65,535 8-bit characters
Type:	character string	VMS data type; symbolic code DSC$K_ DTYPE_T
Access:	read only	Data must be read as input by NOTES$NOTES and not modified
Mechanism:	by reference	Address of data is passed to NOTES$NOTES

If you do not supply an argument, no special processing will be done and VAX Notes will start up in the default Notebook class, as if the command were $ NOTES.

Description

The NOTES$NOTES routine invokes the interactive VAX Notes product, with the appropriate user interface based on the input device, and processes commands from SYS$INPUT until it returns to the calling program. See the <REFERENCE>(VAX_NOTES_GT) for more information on this routine.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_BEGIN

Creates and initializes the profile context for later calls to NOTES$PROFILE_xxx routines.

Format

```
NOTES$PROFILE_BEGIN  (context,input-item-list
                          [,output-item-list])
```

Arguments

context
The address of a longword that will receive profile context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$PROFILE_BEGIN routine creates and initializes the profile context for later calls to NOTES$PROFILE_xxx routines.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_END

Deletes or deallocates the profile context.

Format

```
NOTES$PROFILE_END   (context[,input-item-list]
                         [,output-item-list])
```

Arguments

context
The address of a longword that contains the profile context returned by NOTES$PROFILE_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$PROFILE_END routine deletes or deallocates the profile context. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_GET

Gets the current settings of the personal profile information.

Format

```
NOTES$PROFILE_GET   (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the profile context returned by NOTES$PROFILE_BEGIN.

input-item-list
An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_PROFILE_TEMPORARY

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$PROFILE_GET routine gets the current settings of the personal profile information.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_GET

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$PROFILE_MODIFY

Sets or changes personal profile information.

Format

```
NOTES$PROFILE_MODIFY   (context,input-item-list
                                    [,output-item-list])
```

Arguments

context

The address of a longword that contains the profile context returned by NOTES$PROFILE_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL
NOTES$K_PROFILE_AUTO_DIR
NOTES$K_PROFILE_AUTO_UNSEEN
NOTES$K_PROFILE_CLASS_NAME
NOTES$K_PROFILE_EDITOR
NOTES$K_PROFILE_EDITOR_SPAWN
NOTES$K_PROFILE_PEN_NAME
NOTES$K_PROFILE_PRINT
NOTES$K_PROFILE_TEMPORARY

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_ADD

Verifies that there is no existing user with the specified user name and node names, and creates a member record for the user.

Format

```
NOTES$USER_ADD  (context,input-item-list
                        [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$USER_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_USER_NAME

Optional:

NOTES$K_NOSIGNAL
NOTES$K_USER_ACCESS_LIST
NOTES$K_USER_CREATE_KEYWORD
NOTES$K_USER_MAIL_ADDR
NOTES$K_USER_MODERATE
NOTES$K_USER_NODENAME
NOTES$K_USER_WRITE_BYPASS

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_ADD routine verifies that there is no existing user with the specified user name and node names, and creates a member record for the user.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_BEGIN

Creates and initializes the user context for later calls to NOTES$USER_xxx routines.

Format

```
NOTES$USER_BEGIN  (context,input-item-list
                        [,output-item-list])
```

Arguments

context
The address of a longword that will receive user context information.

input-item-list
An item list selected from the following:

Required:

NOTES$K_NOTEFILE_CONTEXT

Optional:

NOTES$K_NOSIGNAL

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_BEGIN routine creates and initializes the user context for later calls to NOTES$USER_xxx routines.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_BEGIN

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_DELETE

Verifies that the specified user exists, and deletes the member record for that user.

Format

```
NOTES$USER_DELETE   (context,input-item-list
                                [,output-item-list])
```

Arguments

context
The address of a longword that contains the note context returned by NOTES$USER_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_USER_NAME

Optional:

NOTES$K_NOSIGNAL
NOTES$K_USER_ACCESS_LIST
NOTES$K_USER_NODENAME

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_DELETE routine verifies that the specified user exists, and deletes the member record for that user.

Return Code	Description	Status
NOTES$_?		

NOTES$USER_DELETE

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_END

Deletes or deallocates the user context.

Format

```
NOTES$USER_END   (context[,input-item-list]
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$USER_BEGIN.

input-item-list

An item list selected from the following:

Required:

None

Optional:

NOTES$K_NOSIGNAL

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_END routine deletes or deallocates the user context. The context longword is zeroed.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_END

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_GET

Gets information about users.

Format

```
NOTES$USER_GET  (context,input-item-list
                        [,output-item-list])
```

Arguments

context
The address of a longword that contains the note context returned by NOTES$USER_BEGIN.

input-item-list
An item list selected from the following:

Required:

NOTES$K_USER_NAME

Optional:

NOTES$K_CONTINUE
NOTES$K_NOSIGNAL
NOTES$K_USER_MODERATE

output-item-list
An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_GET routine gets information about users.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		

NOTES$USER_GET

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

NOTES$USER_MODIFY

Verifies that the specified user exists, and modifies the member record for that user.

Format

```
NOTES$USER_MODIFY   (context,input-item-list
                                [,output-item-list])
```

Arguments

context

The address of a longword that contains the note context returned by NOTES$USER_BEGIN.

input-item-list

An item list selected from the following:

Required:

NOTES$K_USER_NAME

Optional:

NOTES$K_NOSIGNAL
NOTES$K_USER_ACCESS_LIST
NOTES$K_USER_CREATE_KEYWORD
NOTES$K_USER_MAIL_ADDR
NOTES$K_USER_MODERATE
NOTES$K_USER_NEW_NAME
NOTES$K_USER_NEW_NODENAME
NOTES$K_USER_NODENAME
NOTES$K_USER_WRITE_BYPASS

output-item-list

An item list selected from the valid item codes for output item lists in Section 2.1.

Description

The NOTES$USER_MODIFY routine verifies that the specified user exists, and modifies the member record for that user.

Return Code	Description	Status
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		
NOTES$_?		

Part III

Appendixes

A

Item List Format and Contents

Many of the VAX Notes routines accept input parameters by means of item lists rather than coding a multitude of parameters. The item list format is identical to that used by many of the VMS system services (for example, $GETJPI or $GETDVI). The items in the list may be specified in any order.

Numeric values for the item codes are supplied with the software kit in files named NOTES$ITEMDEF.*; therefore, the numeric values are not defined in this document. The NOTES$ITEMDEF.* files also include values for error codes.

A.1 Various Context Items

These items are used to pass the address of a longword containing context information. Before calling a NOTES$xxx_BEGIN routine, the value of this longword should be set to zero; the routine will store a nonzero context value here, which can be passed to other operations. The longword must not be modified by the caller. A call to the associated NOTES$xxx_END routine will release this context, and reset the longword to zero.

- NOTES$K_CLASS_CONTEXT - Passes the CLASS context to other operations.
- NOTES$K_ENTRY_CONTEXT - Passes the ENTRY context to other operations.
- NOTES$K_KEYWORD_CONTEXT - Passes the KEYWORD context to other operations.
- NOTES$K_NOTE_CONTEXT - Passes the NOTE context to other operations.
- NOTES$K_NOTEFILE_CONTEXT - Passes the NOTEFILE context to other operations.
- NOTES$K_PROFILE_CONTEXT - Passes the PROFILE context to other operations.

- NOTES$K_SERVER_CONTEXT - Passes the SERVER context to other operations.
- NOTES$K_USER_CONTEXT - Passes the USER context to other operations.

A.2 Item Codes

The codes described in Table A–1 are specified as parameters to the NOTES$ callable routines. See the files named NOTES$ITEMDEF.*, supplied with the software kit, for the values assigned to these codes.

Table A–1 Item Codes

Item Code	Description
NOTES$K_CLASS_HINT_GET_ENTRY	Indicates that a NOTES$CLASS_ GET_ENTRY will be performed next (improves server performance)
NOTES$K_CLASS_NAME	Class name string (wildcards may be allowed, depending on the context)
NOTES$K_CLASS_NEW_NAME	New class name string (renames the old class)
NOTES$K_CLASS_USER_AREA	String of user-programmable data
NOTES$K_CONTINUE	Keeps returning information
NOTES$K_DELIF0	Delete class if empty, indicated by 1
NOTES$K_ENTRY_HINT_GET_CLASS	Indicates that a NOTES$ENTRY_ GET_CLASS will be performed next (improves server performance)
NOTES$K_ENTRY_HINT_GET_KEYWORD	Indicates that a NOTES$ENTRY_GET_ KEYWORD will be performed next (improves server performance)
NOTES$K_ENTRY_LAST_STATUS	Longword value of the last update for the entry (update status longword)
NOTES$K_ENTRY_NAME	Entry name string (wildcards may be allowed, depending on the context)

(continued on next page)

Table A–1 (Cont.) Item Codes

Item Code	Description
NOTES$K_ENTRY_NEW_NAME	New entry name string (renames the old entry)
NOTES$K_ENTRY_OBJECT_NAME	DNS object name string for the entry
NOTES$K_ENTRY_OBJECT_SPEC	File specification string to which the DNS object name was translated
NOTES$K_ENTRY_UNSEEN_EST	Longword number of unseen notes
NOTES$K_ENTRY_USER_AREA	String of user-programmable data
NOTES$K_KEYWORD_HINT_GET_NOTE	Indicates that a NOTES$KEYWORD_GET_NOTE will be performed next (improves server performance)
NOTES$K_KEYWORD_NAME	Keyword name string (wildcards may be allowed, depending on the context)
NOTES$K_KEYWORD_NEW_NAME	New keyword name string (renames the old keyword)
NOTES$K_KEYWORD_USER_AREA	String of user-programmable data
NOTES$K_NOSIGNAL	Overrides default error signalling
NOTES$K_NOTE_ALL_RESPONSES	Indicates replies should also be returned
NOTES$K_NOTE_AUTHOR	Author name string
NOTES$K_NOTE_BACK_NOTE	Selects the previous topic
NOTES$K_NOTE_BACK_RESPONSE	Selects the previous reply
NOTES$K_NOTE_BEFORE_TIME	Selects only notes before this quadword date/time
NOTES$K_NOTE_BLINK_ID	Note identifier string
NOTES$K_NOTE_BLINK_TITLE	Topic title string
NOTES$K_NOTE_BLINK_UID	Longword unique note identifier
NOTES$K_NOTE_CAN_REPLY	Longword indicating with a value of 1 that the caller can reply
NOTES$K_NOTE_CREATE_TIME	Quadword date/time of note creation

(continued on next page)

Table A–1 (Cont.) Item Codes

Item Code	Description
NOTES$K_NOTE_HIDDEN	Selects or modifies hidden notes. For selection, selects hidden notes = 1, selects unhidden notes = 0; for modifying a note, hide note = 1, unhide note = 0
NOTES$K_NOTE_HINT_GET_KEYWORD	Value of 1 indicates that a call to NOTES$GET_KEYWORD is next
NOTES$K_NOTE_HINT_GET_TEXT	Value of 1 indicates that a call to the NOTES$GET_TEXT routine is next
NOTES$K_NOTE_ID	String of the note identifier or a range of notes
NOTES$K_NOTE_MARK_SEEN	Mark notes as seen, indicated by 1
NOTES$K_NOTE_NEW_ID	String specifying the new note-ID to which the range of notes should be moved
NOTES$K_NOTE_NEXT_NOTE	Selects the next topic
NOTES$K_NOTE_NEXT_RESPONSE	Select the next reply
NOTES$K_NOTE_NEXT_UNSEEN	Select the next unseen note
NOTES$K_NOTE_NUMRECORDS	Longword number of text records in this note
NOTES$K_NOTE_NUMRESPONSES	Longword response number of last reply
NOTES$K_NOTE_PEN_NAME	Personal name string
NOTES$K_NOTE_SEARCH_STRING	Selects only notes containing this search string
NOTES$K_NOTE_SINCE_TIME	Selects only notes after this quadword date/time
NOTES$K_NOTE_TITLE	Title string
NOTES$K_NOTE_TYPE	Longword indicating the type of text in the note
NOTES$K_NOTE_UID	Longword unique note identifier
NOTES$K_NOTE_UNSEEN	Selects notes; select unseen notes = 1, select seen notes = 0
NOTES$K_NOTE_USER_AREA	String of user-programmable data

(continued on next page)

Table A–1 (Cont.) Item Codes

Item Code	Description
NOTES$K_NOTE_WRITELOCK	Modifies locking of notes; writelock note = 1, remove writelock on note = 0
NOTES$K_NOTEFILE_CREATE	Creates the specified notefile
NOTES$K_NOTEFILE_CREATE_TIME	Quadword date/time of the notefile creation
NOTES$K_NOTEFILE_DEFAULT_NAME	Default notefile name string
NOTES$K_NOTEFILE_ENTRYTOTAL	Total number of topics and replies
NOTES$K_NOTEFILE_FILE_NAME	Notefile name string
NOTES$K_NOTEFILE_FORMAT	Longword indicating notefile format
NOTES$K_NOTEFILE_HIGH_UID	Longword of highest used note UID
NOTES$K_NOTEFILE_LASTREV	Quadword date/time of the last notefile revision
NOTES$K_NOTEFILE_MODERATE	Changes moderator status for the notefile; attain moderator status = 1, cancel moderator status = 0
NOTES$K_NOTEFILE_MODERATOR	Moderator name string
NOTES$K_NOTEFILE_NOTICE	Notice string
NOTES$K_NOTEFILE_NUMNOTES	Longword note number of the last topic in the notefile
NOTES$K_NOTEFILE_RELATED_NAME	Related name string for the notefile
NOTES$K_NOTEFILE_REPLY_ONLY	Conference in which only authorized users may enter topics, indicated by 1
NOTES$K_NOTEFILE_RESTRICTED	Modifies or determines whether user access is restricted; only authorized users may participate = 1, any user may participate = 0
NOTES$K_NOTEFILE_RESULT_SPEC	Resultant file name specification string of the notefile
NOTES$K_NOTEFILE_TITLE	Notefile title string
NOTES$K_NOTEFILE_USER_AREA	String of user-programmable data
NOTES$K_NOTEFILE_WRITELOCK	Notefile writelocked against new notes, indicated by 1

(continued on next page)

Table A–1 (Cont.) Item Codes

Item Code	Description
NOTES$K_PROFILE_AUTO_DIR	Modifies or determines the automatic directory setting; automatic directory = 1, no automatic directory = 0
NOTES$K_PROFILE_AUTO_UNSEEN	Modifies or determines the automatic unseen setting; automatic unseen = 1, no automatic unseen = 0
NOTES$K_PROFILE_CLASS_NAME	String of default class name to use
NOTES$K_PROFILE_EDITOR	Editor name string
NOTES$K_PROFILE_EDITOR_SPAWN	Flag to indicate whether the editor must be spawned; Spawn editor = 1, callable editor = 0
NOTES$K_PROFILE_PEN_NAME	Personal name string
NOTES$K_PROFILE_PRINT	String of default print qualifiers to apply; allows the user to define defaults for the PRINT command
NOTES$K_PROFILE_TEMPORARY	Indicates that the specified profile settings are temporary, and should not be stored
NOTES$K_SEEN_MAP	String indicating which notes have been seen. User programs may not interpret this byte string, except that a null string indicates that no notes have been seen
NOTES$K_SERVER_MAX_ITEM	Longword value of the highest item code that the server supports
NOTES$K_TEXT_END	Indicates the end of the note text for a note being added
NOTES$K_TEXT_STRING	Text record string in a note
NOTES$K_TEXT_TYPE	Longword type of the text record

(continued on next page)

Table A–1 (Cont.) Item Codes

Item Code	Description
NOTES$K_UNIENTRY	Unique by entry (for markers)
NOTES$K_USER_ACCESS_LIST	String of node::user-name pairs for this user
NOTES$K_USER_CREATE_KEYWORD	For a conference, modifies or determines whether any user may create keywords: only users with CREATE_KEYWORDS privilege may create keywords = 1 (default), any user may create keywords = 0; for a member, determines whether the member has CREATE_KEYWORDS privilege: may create keywords = 1, may not create keywords = 0 (default)
NOTES$K_USER_MAIL_ADDR	User's mailing address string
NOTES$K_USER_MODERATE	Modifies or determines whether user may moderate the notefile; may moderate = 1, may not moderate = 0
NOTES$K_USER_NAME	User name string (wildcards may be allowed, depending on the context)
NOTES$K_USER_NEW_NAME	New user name string (renames the user)
NOTES$K_USER_NEW_NODENAME	Changes the node names string
NOTES$K_USER_NODENAME	Node names string (comma-separated), which, together with the user name, indicates which processes may access this (private) notefile
NOTES$K_USER_WRITE_BYPASS	User may write to write-locked conference, reply-only conference, and write-locked topic

B
Sample Program

The BASIC program presented in this appendix extracts unseen notes from the notefile (or notefiles) specified interactively by the user, and writes the note headings and text to a VMS text file. Each specified notefile must be an entry in the user's Notebook. The entire notefile is checked for unseen notes; ranges of notes cannot be specified.

There are five parts to this program:

1. ITEMDEF.BAS—Definitions used for creating item lists
2. NOTEDEF.BAS—Definitions of callable routine entrypoints
3. NOTES$ITEMDEF.BAS—Definitions of constants for callable routine
4. EXTRACT_NOTE.BAS—Main program to extract unseen notes
5. ITEM_LIST.BAS—Subprogram to enter items into item lists

In ITEMDEF.BAS, separate item list formats are set up for an empty item list (no_list), an item list with only one item in it (one_list), an input item list (inp_list), and an output item list (out_list). This is required for specifying input and output item lists for the callable routines. The inp-list and out-list item lists allow a maximum of 15 entries to be passed. The no-list and one-list item lists are for those routines that do not require more than one parameter.

The definition file NOTES$ITEMDEF.BAS, which is provided with the software kit, contains the definitions (numeric values) of the item codes. Definition files for other languages are also provided with the software kit; following is the complete list:

- NOTES$ITEMDEF.ADA
- NOTES$ITEMDEF.BAS
- NOTES$ITEMDEF.FOR
- NOTES$ITEMDEF.H
- NOTES$ITEMDEF.MAR

Sample Program

- NOTES$ITEMDEF.PAS
- NOTES$ITEMDEF.PLI
- NOTES$ITEMDEF.R32
- NOTES$ITEMDEF.SDL

Only the item codes used by the EXTRACT_NOTE.BAS program are included in the sample program.

The part NOTEDEF.BAS contains definitions for the error codes used in the EXTRACT_NOTE.BAS program. A complete list of error codes and their values is provided in the file NOTES$ITEMDEF.BAS.

In the main program EXTRACT_NOTE.BAS, it is important to note the order in which the VAX Notes routines are called. First, the NOTES$NOTEFILE_ BEGIN routine is called to open the notefile specified by the entry name supplied by the user (or the first entry name matching the user's input if that input included a wildcard). As long as the notefile remains open, any work can be performed on the data in that notefile. In the program EXTRACT_NOTE.BAS, this work consists only of extracting unseen notes. The NOTES$NOTEFILE_BEGIN routine is called in the segment beginning with line 1104.

In the segment beginning with line 1106, the NOTES$NOTE_GET routine is called to make sure there are unseen notes to get. If so, an output file (NOTES.TXT) is opened, and the conference heading information is formatted and written to the output file.

A loop is set up within segments beginning with lines 1108 and 1109 to get all the text for all unseen notes in the currently open notefile. The NOTES$NOTE_GET routine returns only the note heading information. In the segment beginning with line 1109, the NOTES$NOTE_GET_TEXT is called repeatedly for each line of text for the note retrieved with NOTES$NOTE_GET. (Note text is retrieved one record or line at a time.) Once all the text for the note is extracted and written to the output file (NOTE.TXT), the program gets the next unseen note and repeats this process until all unseen notes in the file have been extracted.

Finally, the notefile is closed (line 1202), and the program checks to see whether the user specified another notefile. If so, the loop starting at line 1104 is executed again. If not, the program ends.

You must create a command procedure for linking as follows:

```
$ LINK yourfile, SYS$INPUT:/OPTION
  SYS$LIBRARY:NOTES$SHARE/SHARE
$ EXIT
```

Example B–1 Sample BASIC Program

```
------------------------------------------------------------------------------
ITEMDEF.BAS
------------------------------------------------------------------------------

        DECLARE BYTE CONSTANT max_item_index = 14
        DECLARE BYTE CONSTANT max_item_entries = (max_item_index+1%)*3%

        !----------------------------------------------------------------!
        !                                                                !
        ! Common RECORD descriptor for an item list of 15 items          !
        !                                                                !
        !----------------------------------------------------------------!
        RECORD ITEM_LIST_DESCRIPTOR
          GROUP item (max_item_index)
            VARIANT
            CASE
                WORD buffer_length
                WORD item_code
                LONG buffer_address
                LONG return_length_address
            CASE
                LONG terminator
            END VARIANT
          END GROUP
          LONG FILL
        END RECORD

        !----------------------------------------------------------------!
        !                                                                !
        ! Common RECORD descriptor for a single item list                !
        !                                                                !
        !----------------------------------------------------------------!
        RECORD ONE_LIST_DESCRIPTOR
                WORD buffer_length
                WORD item_code
                LONG buffer_address
                LONG return_length_address
                LONG FILL
        END RECORD
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
!----------------------------------------------------------------------!
!                                                                      !
! MAP command for sharing of memory                                    !
!                                                                      !
!----------------------------------------------------------------------!

MAP (itemls)    LONG no_list,                                  &
                ONE_LIST_DESCRIPTOR one_list,                  &
                ITEM_LIST_DESCRIPTOR inp_list,                 &
                ITEM_LIST_DESCRIPTOR out_list
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
------------------------------------------------------------------------------
NOTEDEF.BAS
------------------------------------------------------------------------------

        %INCLUDE "NOTEITEMS"

        DECLARE LONG CONSTANT                                   &
                        NOTES$_NO_SUCH_NOTE = 66830642,         &
                        NOTES$_NO_MORE_NOTES = 66822176,        &
                        NOTES$_NO_MORE_ENTRIES = 66822160,      &
                        NOTES$_NO_MORE_TEXT = 66822184

        EXTERNAL LONG FUNCTION                                  &
                        NOTES$NOTEFILE_BEGIN,                   &
                        NOTES$NOTEFILE_END,                     &
                                                                &
                        NOTES$ENTRY_BEGIN,                      &
                        NOTES$ENTRY_GET,                        &
                        NOTES$ENTRY_END,                        &
                                                                &
                        NOTES$NOTE_BEGIN,                       &
                        NOTES$NOTE_GET,                         &
                        NOTES$NOTE_GET_TEXT,                    &
                        NOTES$NOTE_END

------------------------------------------------------------------------------
NOTEITEMS.BAS
------------------------------------------------------------------------------

declare long constant NOTES$K_NOSIGNAL = 1
!  Setting this on a BEGIN operation disables signals
! ----------------------------*/
declare long constant NOTES$K_TEXT_STRING = 2
! ----------------------------*/
declare long constant NOTES$K_NOTE_AUTHOR = 6
declare long constant NOTES$K_NOTE_BLINK_ID = 10
declare long constant NOTES$K_NOTE_CREATE_TIME = 12
declare long constant NOTES$K_NOTE_ID = 14
declare long constant NOTES$K_NOTE_NUMRECORDS = 17
declare long constant NOTES$K_NOTE_NUMRESPONSES = 18
declare long constant NOTES$K_NOTE_PEN_NAME = 19
declare long constant NOTES$K_NOTE_TITLE = 23
declare long constant NOTES$K_NOTE_UNSEEN = 25
! ----------------------------*/
```

(continued on next page)

Sample Program

Example B–1 (Cont.) Sample BASIC Program

```
declare long constant NOTES$K_NOTEFILE_CONTEXT = 28
! ----------------------------*/
declare long constant NOTES$K_CONTINUE = 38
! ----------------------------*/
declare long constant NOTES$K_ENTRY_NAME = 39
! ----------------------------*/
declare long constant NOTES$K_NOTEFILE_CREATE_TIME = 46
declare long constant NOTES$K_NOTEFILE_DEFAULT_NAME = 47
declare long constant NOTES$K_NOTEFILE_ENTRYTOTAL = 48
declare long constant NOTES$K_NOTEFILE_FILE_NAME = 49
declare long constant NOTES$K_NOTEFILE_LASTREV = 52
declare long constant NOTES$K_NOTEFILE_NOTICE = 55
declare long constant NOTES$K_NOTEFILE_NUMNOTES = 56
declare long constant NOTES$K_NOTEFILE_RESULT_SPEC = 59
declare long constant NOTES$K_NOTEFILE_TITLE = 60
! ----------------------------*/
declare long constant NOTES$K_SEEN_MAP = 71
! ----------------------------*/
declare long constant NOTES$K_NOTE_BLINK_TITLE = 79
declare long constant NOTES$K_NOTE_HINT_GET_TEXT = 82
! ----------------------------*/

------------------------------------------------------------------------------
EXTRACT_NOTE.BAS
------------------------------------------------------------------------------

1       !
        ! Sample program using the VAX Notes callable routines to
        ! extract all UNSEEN notes from an ENTRY in your NOTEBOOK.
        !
        OPTION TYPE = EXPLICIT

        %INCLUDE "ITEMDEF"            ! Item list RECORD definitions
        %INCLUDE "NOTEDEF"            ! VAX Notes entry points and constants

        EXTERNAL LONG CONSTANT SS$_NORMAL
        EXTERNAL LONG FUNCTION SYS$ASCTIM, SYS$GETMSG

        MAP (cnstnt)    STRING  zero$0                  = 4,            &
                                plus$1                  = 4,            &
                                minus$1                 = 4

        MAP (cnstnt)    LONG    zero,                   ! binary  0%    &
                                plus_1,                 ! binary  1%    &
                                minus_1                 ! binary -1%
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
        MAP (contxt)     STRING  note_ctx               = 4,            &
                                 entry_ctx              = 4,            &
                                 notefile_ctx           = 4,            &
                                 notebook_ctx           = 4
```

(continued on next page)

Sample Program

Example B–1 (Cont.) Sample BASIC Program

```
MAP (contxt)    LONG    note_l_ctx,                         &
                        entry_l_ctx,                        &
                        notefile_l_ctx,                     &
                        notebook_l_ctx

MAP (integr)    STRING  notefile_entrytotal     = 4,        &
                        notefile_numnotes       = 4,        &
                        note_numresponses       = 4,        &
                        note_numrecords         = 4,        &
                        note_type               = 4

MAP (integr)    LONG    notefile_l_entrytotal,              &
                        notefile_l_numnotes,                &
                        note_l_numresponses,                &
                        note_l_numrecords,                  &
                        note_l_type

MAP (stryng)    STRING  notefile_create_time    = 8,        &
                        notefile_lastrev_time   = 8,        &
                        note_create_time        = 8,        &
                  WORD  notefile_result_spec_len,           &
                STRING  notefile_result_spec    = 254,      &
                  WORD  notefile_title_len,                 &
                STRING  notefile_title          = 80,       &
                  WORD  notefile_notice_len,                &
                STRING  notefile_notice         = 80,       &
                  WORD  notefile_file_name_len,             &
                STRING  notefile_file_name      = 254,      &
                  WORD  notefile_seen_map_len,              &
                STRING  notefile_seen_map       = 2048,     &
                  WORD  note_author_len,                    &
                STRING  note_author             = 32,       &
                  WORD  note_pen_name_len,                  &
                STRING  note_pen_name           = 64,       &
                  WORD  note_title_len,                     &
                STRING  note_title,             = 80,       &
                  WORD  note_blink_title_len,               &
                STRING  note_blink_title        = 80,       &
                  WORD  note_id_len,                        &
                STRING  note_id                 = 16,       &
                  WORD  note_blink_id_len,                  &
                STRING  note_blink_id           = 16,       &
                  WORD  note_text_len,                      &
                STRING  note_text               = 512
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
MAP (scrtch)      LONG  ret_status, junk_status,                &
                  WORD  msg_text_len,                           &
                STRING  msg_text                = 256

DECLARE BYTE    inp_index, out_index

DECLARE WORD    i, default_len

DECLARE LONG    notebook_open_flag, output_open_flag

DECLARE STRING                                                  &
                last_note_id,                                   &
                left_part, mid_part, right_part,                &
                entry_name, note_range,                         &
                notebook_name, notebook_default, notefile_default

DECLARE BYTE CONSTANT                                           &
                line_len = 80,                                  &
                inp_initial = 1,                                &
                out_initial = 16

DECLARE STRING CONSTANT                                         &
                notes$notebook = "NOTES$NOTEBOOK",              &
                notes$notebook_default = "SYS$LOGIN:.NOTE",     &
                notes$notefile_default = "NOTES$LIBRARY:.NOTE"

DECLARE LONG FUNCTION                                           &
                fnREPORT_ERROR (LONG)

DECLARE STRING FUNCTION                                         &
                fnCENTER_LINE (STRING),                         &
                fnLEFT (STRING,STRING),                         &
                fnRIGHT (STRING,STRING),                        &
                fnMID (STRING,STRING,STRING),                   &
                fnUNQUOTE (STRING),                             &
                fnPLURAL (LONG,STRING,STRING,STRING),           &
                fnASCII_DATE (STRING),                          &
                fnFORMAT_HEADING (STRING,STRING,STRING)
```

(continued on next page)

Sample Program

Example B–1 (Cont.) Sample BASIC Program

```
1000    ! Initialize error handler and useful constants
        ON ERROR GOTO 19000
        zero = 0%
        plus_1 = 1%
        minus_1 = -plus_1
        notebook_name = notes$notebook
        notebook_default = notes$notebook_default
        notefile_default = notes$notefile_default
        note_range = "*.*"
        default_len = minus_1

1100    ! Main loop to request entry name(s)
        PRINT "Enter entry name";
        LINPUT entry_name
        entry_name = fnUNQUOTE(entry_name)
        GOTO 19020 UNLESS LEN(entry_name)

1101    ! First open notebook to find which entry
        IF NOT notebook_open_flag THEN
                inp_index = inp_initial
                CALL ITEM_LIST (inp_index,notes$k_nosignal,plus$1,default_len)
                CALL ITEM_LIST (inp_index,notes$k_notefile_file_name,notebook_name,default_len)
                CALL ITEM_LIST (inp_index,notes$k_notefile_default_name,notebook_default    &
                                        ,default_len)
                ret_status = NOTES$NOTEFILE_BEGIN (notebook_l_ctx,inp_list,no_list)
                GOTO 2005 UNLESS (ret_status AND ss$_normal)
        ! Prepare for getting notebook entry info
                CALL ITEM_LIST (minus_1,notes$k_notefile_context,notebook_ctx,default_len)
                ret_status = NOTES$ENTRY_BEGIN (entry_l_ctx,one_list,no_list)
                notebook_open_flag = minus_1
        END IF

1103    ! Get notebook entry info
        inp_index = inp_initial
        CALL ITEM_LIST (inp_index,notes$k_nosignal,plus$1,default_len)
        CALL ITEM_LIST (inp_index,notes$k_entry_name,entry_name,default_len)
        out_index = out_initial
        CALL ITEM_LIST (out_index,notes$k_notefile_file_name                                    &
                                        ,notefile_file_name,notefile_file_name_len)
        CALL ITEM_LIST (out_index,notes$k_seen_map,notefile_seen_map,notefile_seen_map_len)
        ret_status = NOTES$ENTRY_GET (entry_l_ctx,inp_list,out_list)
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
1104    ! Opening entry notefile
        WHILE (ret_status AND ss$_normal)
            PRINT "Extracting UNSEEN notes from "                                    &
                                + LEFT$(notefile_file_name,notefile_file_name_len)
            inp_index = inp_initial
            CALL ITEM_LIST (inp_index,notes$k_nosignal,plus$1,default_len)
            CALL ITEM_LIST (inp_index,notes$k_notefile_default_name,notefile_default,default_len)
            CALL ITEM_LIST (inp_index,notes$k_notefile_file_name,notefile_file_name   &
                                            ,notefile_file_name_len)
            CALL ITEM_LIST (inp_index,notes$k_seen_map,notefile_seen_map,notefile_seen_map_len)
            out_index = out_initial
            CALL ITEM_LIST (out_index,notes$k_notefile_create_time,notefile_create_time  &
                                            ,default_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_lastrev,notefile_lastrev_time,default_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_entrytotal,notefile_entrytotal,default_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_numnotes,notefile_numnotes,default_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_result_spec,notefile_result_spec  &
                                            ,notefile_result_spec_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_title,notefile_title,notefile_title_len)
            CALL ITEM_LIST (out_index,notes$k_notefile_notice,notefile_notice,notefile_notice_len)
            ret_status = NOTES$NOTEFILE_BEGIN (notefile_l_ctx,inp_list,out_list)
            GOTO 2005 UNLESS (ret_status AND ss$_normal)
1105    ! Prepare for reading notes
            CALL ITEM_LIST (minus_1,notes$k_notefile_context,notefile_ctx,default_len)
            ret_status = NOTES$NOTE_BEGIN (note_l_ctx,one_list,no_list)
            GOTO 1202 UNLESS (ret_status AND ss$_normal)
1106    ! Now we can start getting notes
            inp_index = inp_initial
            CALL ITEM_LIST (inp_index,notes$k_nosignal,plus$1,default_len)
            CALL ITEM_LIST (inp_index,notes$k_note_id,note_range,default_len)
            CALL ITEM_LIST (inp_index,notes$k_note_unseen,plus$1,default_len)
            CALL ITEM_LIST (inp_index,notes$k_note_hint_get_text,plus$1,default_len)
            out_index = out_initial
            CALL ITEM_LIST (out_index,notes$k_note_id,note_id,note_id_len)
            CALL ITEM_LIST (out_index,notes$k_note_author,note_author,note_author_len)
            CALL ITEM_LIST (out_index,notes$k_note_pen_name,note_pen_name,note_pen_name_len)
            CALL ITEM_LIST (out_index,notes$k_note_title,note_title,note_title_len)
            CALL ITEM_LIST (out_index,notes$k_note_create_time,note_create_time,default_len)
            CALL ITEM_LIST (out_index,notes$k_note_numresponses,note_numresponses,default_len)
            CALL ITEM_LIST (out_index,notes$k_note_numrecords,note_numrecords,default_len)
            CALL ITEM_LIST (out_index,notes$k_note_blink_id,note_blink_id,note_blink_id_len)
            CALL ITEM_LIST (out_index,notes$k_note_blink_title,note_blink_title       &
                                            ,note_blink_title_len)
            ret_status = NOTES$NOTE_GET (note_l_ctx,inp_list,out_list)
            GOTO 1201 if (ret_status = notes$_no_such_note)
            last_note_id = LEFT$(note_id,note_id_len)
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
1107    ! Open output file for extracts (if not already opened)
           IF NOT output_open_flag THEN
                OPEN "NOTE.TXT" FOR OUTPUT AS FILE #3%                &
                               ,SEQUENTIAL VARIABLE                   &
                               ,ALLOW NONE                            &
                               ,ACCESS SCRATCH                        &
                               ,RECORDSIZE 512%                       &
                               ,DEFAULTNAME "SYS$LOGIN:NOTE.TXT"
                output_open_flag = minus_1
           ELSE PRINT #3%, FF
           END IF
         ! format notefile header
           PRINT #3%, fnCENTER_LINE("<<< "     &
                               + LEFT$(notefile_result_spec,notefile_result_spec_len) + " >>>")
           PRINT #3%, fnCENTER_LINE("-< " + LEFT$(notefile_title,notefile_title_len) + ">-")
           PRINT #3%, fnFORMAT_HEADING("Created: " + fnASCII_DATE(notefile_create_time) &
                               ,fnPLURAL(notefile_l_numnotes,"topic","s","")   &
                               ,"Updated: " + fnASCII_DATE(notefile_lastrev_time))
           PRINT #3%, fnCENTER_LINE("-< "      &
                               + LEFT$(notefile_notice,notefile_notice_len) + " >-") &
                                                  IF (notefile_notice_len)
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
1108     ! Continue reading notes until no more
             WHILE (ret_status <> notes$_no_more_notes)
                IF (fnLEFT(last_note_id,".") <> fnLEFT(note_id,".")) THEN
                      PRINT #3%, FF
                ELSE    PRINT #3%
                END IF
                last_note_id = LEFT$(note_id,note_id_len)
           ! Format note header
                PRINT #3%, STRING$(line_len,ASCII("="))
                IF (POS(last_note_id,".0",1%) = zero) THEN
                        right_part = fnRIGHT(last_note_id,".") + " of "          &
                                                + NUM1$(note_l_numresponses)
                        mid_part = LEFT$(note_blink_title,note_blink_title_len)
                ELSE    mid_part = LEFT$(note_title,note_title_len)
                        note_title_len = zero
                        IF (note_l_numresponses = zero) THEN
                                right_part = "No replies"
                        ELSE    right_part = fnPLURAL(note_l_numresponses,"repl","ies","y")
                        END IF
                END IF
                PRINT #3%, fnFORMAT_HEADING("Note "+last_note_id,mid_part,right_part)
                left_part = LEFT$(note_author,note_author_len)
                left_part = left_part + ' "' + LEFT$(note_pen_name,note_pen_name_len) + '"' &
                                                    IF (note_pen_name_len)
                PRINT #3%, fnFORMAT_HEADING(left_part,SP                 &
                                ,fnPLURAL(note_l_numrecords,"line","s","") &
                                + "  " + fnASCII_DATE(note_create_time))
```

(continued on next page)

Sample Program

Example B–1 (Cont.) Sample BASIC Program

```
                PRINT #3%, STRING$(line_len,ASCII("-"))
                PRINT #3%, fnCENTER_LINE("-< " + LEFT$(note_title,note_title_len) + ">-") &
                                                IF (note_title_len)
                PRINT #3%

                inp_index = inp_initial
                CALL ITEM_LIST (inp_index,notes$k_continue,plus$1,default_len)
                CALL ITEM_LIST (zero,notes$k_text_string,note_text,note_text_len)
                ret_status = NOTES$NOTE_GET_TEXT (note_l_ctx,inp_list,one_list)

1109    ! Continue reading the text of the note until no more
                WHILE (ret_status <> notes$_no_more_text)
                        PRINT #3%, LEFT$(note_text,note_text_len)
                        ret_status = NOTES$NOTE_GET_TEXT (note_l_ctx,inp_list,one_list)
                NEXT
                ret_status = NOTES$NOTE_GET (note_l_ctx,inp_list,out_list)
            NEXT

1201    ! Finish with note context
            junk_status = NOTES$NOTE_END (note_l_ctx,no_list,no_list)

1202    ! Close the notefile
            junk_status = NOTES$NOTEFILE_END (notefile_l_ctx,no_list,no_list)

1203    ! Try for another wildcard entry
            junk_status = fnREPORT_ERROR(notes$_no_more_notes)
            CALL ITEM_LIST (minus_1,notes$k_continue,plus$1,default_len)
            out_index = out_initial
            CALL ITEM_LIST (out_index,notes$k_notefile_file_name              &
                                    ,notefile_file_name,notefile_file_name_len)
            CALL ITEM_LIST (out_index,notes$k_seen_map,notefile_seen_map,notefile_seen_map_len)
            ret_status = NOTES$ENTRY_GET (entry_l_ctx,one_list,out_list)
        NEXT

2005    ! Output any unexpected error
        junk_status = fnREPORT_ERROR(notes$_no_more_entries)
        GOTO 1100
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
15000   ! User defined functions

        DEF STRING fnCENTER_LINE (STRING x)
        fnCENTER_LINE = SPACE$((line_len-LEN(x)+1%)/2%) + x
        END DEF

        DEF STRING fnLEFT(STRING x, STRING y)
        fnLEFT = LEFT$(x,POS(x,y,1%)-1%)
        END DEF
        DEF STRING fnRIGHT(STRING x, STRING y)
        fnRIGHT = RIGHT$(x,POS(y+x,y,LEN(y)+1%))
        END DEF

        DEF STRING fnMID (STRING x, STRING y, STRING z)
        fnMID = fnLEFT(fnRIGHT(x,y)+z,z)
        END DEF

        DEF STRING fnUNQUOTE (STRING x)
        x = EDIT$(x,4%+32%+8%+16%+128%+256%)
        IF (ASCII(x) = ASCII('"')) OR (ASCII(x) = ASCII("'")) THEN
                x = MID$(x,2%,LEN(x)-2%) IF (ASCII(x)=ASCII(MID$(x,LEN(x),1%)))
        END IF
        fnUNQUOTE = x
        END DEF

        DEF STRING fnPLURAL(LONG n, STRING x, STRING y, STRING z)
        y = z IF (n = 1%)   ! singular
        fnPLURAL = NUM1$(n) + SP + x + y
        END DEF

        DEF STRING fnASCII_DATE(STRING x)
        DECLARE STRING time_string
        time_string = SPACE$(24%)
        junk_status = SYS$ASCTIM(,time_string,x BY REF,)
        fnASCII_DATE = LEFT$(time_string,17%)
        END DEF
        DEF STRING fnFORMAT_HEADING(STRING x, STRING y, STRING z)
        y = fnCENTER_LINE(y)
        x = x + MID$(y,LEN(x)+1%,LEN(y))
        fnFORMAT_HEADING = x + SPACE$(line_len-LEN(x)-LEN(z)) + z
        END DEF

        DEF LONG fnREPORT_ERROR (LONG e)
        IF (ret_status <> e) THEN
                junk_status = SYS$GETMSG(ret_status BY VALUE,msg_text_len,msg_text,15% BY VALUE,)
                PRINT LEFT$(msg_text,msg_text_len) IF (junk_status AND ss$_normal)
        END IF
        END DEF
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
19000 ! Error recovery
        RESUME 19010
19010   PRINT ERR;ERL;'"' + ERT$(ERR) + '"' UNLESS (ERR = 11%)
19020   ! Finish with entry context
        junk_status = NOTES$ENTRY_END (entry_l_ctx,no_list,no_list)
19030   ! Close notebook
        junk_status = NOTES$NOTEFILE_END (notebook_l_ctx,no_list,no_list)
19040   ! Close output file
        CLOSE #3%
32767   END

------------------------------------------------------------------------------
ITEM_LIST.BAS
------------------------------------------------------------------------------

1       SUB ITEM_LIST (BYTE indx, WORD code, STRING entry, WORD rlen)

        OPTION TYPE = EXPLICIT

        %INCLUDE "ITEMDEF"            ! Item list RECORD definitions

        DECLARE BYTE lcl_indx
        DECLARE WORD buff_len
        DECLARE LONG buff_adr, rlen_loc, ret_status

        EXTERNAL LONG FUNCTION STR$ANALYZE_SDESC

        ret_status = STR$ANALYZE_SDESC (entry,buff_len,buff_adr)

        rlen_loc = LOC(rlen) IF (rlen <> -1%)   ! -1 = don't care

        SELECT indx
           CASE -1% TO 0%                       ! -1 = input, 0 = output/input
                IF indx THEN
                       rlen_loc = 0%
                       buff_len = rlen IF (rlen < buff_len) IF (rlen>= 0%)
                ELSE   rlen = 0% IF rlen_loc
                END IF
                one_list::item_code             = code
                one_list::buffer_length         = buff_len
                one_list::buffer_address        = buff_adr
                one_list::return_length_address = rlen_loc
```

(continued on next page)

Example B–1 (Cont.) Sample BASIC Program

```
          CASE 1% TO 15%                          ! input item list
                buff_len = rlen IF (rlen < buff_len) IF (rlen>= 0%)
                lcl_indx = indx - 1%
                inp_list::item(lcl_indx)::item_code             = code
                inp_list::item(lcl_indx)::buffer_length         = buff_len
                inp_list::item(lcl_indx)::buffer_address        = buff_adr
                inp_list::item(lcl_indx)::return_length_address = 0%
                inp_list::item(lcl_indx+1%)::terminator = 0%            &
                                        IF (lcl_indx < max_item_index)
                indx = indx + 1%
          CASE 16% TO 30%                         ! Output item list
                rlen = 0% IF rlen_loc
                lcl_indx = indx - 16%
                out_list::item(lcl_indx)::item_code             = code
                out_list::item(lcl_indx)::buffer_length         = buff_len
                out_list::item(lcl_indx)::buffer_address        = buff_adr
                out_list::item(lcl_indx)::return_length_address = rlen_loc
                out_list::item(lcl_indx+1%)::terminator = 0%            &
                                        IF (lcl_indx < max_item_index)
                indx = indx + 1%
        END SELECT
32767   END SUB
```

C
Messages

(TBS)