HONEYWELL

MULTICS FORUM INTERACTIVE MEETING SYSTEM USER'S GUIDE

SOFTWARE

MULTICS FORUM
INTERACTIVE MEETING SYSTEM
USER'S GUIDE

SUBJECT

Description of the Multics Forum Interactive Meeting System

SPECIAL INSTRUCTIONS

This edition supersedes Revision 0 of the manual dated July 1982 and its associated addendum, Addendum A, dated February 1983. Refer to the Preface for "Significant Changes."

Throughout the manual, change bars in the margins indicate technical changes and additions; asterisks denote deletions.

SOFTWARE SUPPORTED

Multics Software Release 10.2

ORDER NUMBER

CY74-01

December 1983

Honeywell

PREFACE

The software product identified as Multics Forum is the property of the Massachusetts Institute of Technology and distributed by Honeywell Information Systems under license from the Massachusetts Institute of Technology.

This book is a detailed description of the Multics Forum Interactive Meeting System. It is intended for all users; both those who have relatively little experience on the Multics operating system (or any other computer system) and experienced programmers will find this a complete description. You are, however, expected to be familiar with the Multics concepts described in the 2-volume set, *New Users' Introduction to Multics* -- Part I (Order No. CH24), and -- Part II (Order No. CH25), referred to in this manual as New Users' Introduction.

The sections of this manual fully describe how to attend Forum's online meetings. The first five sections tell you everything you need to know to attend meetings. This part of the manual may be all you need if you are a beginning Multics user. Section 6 describes some advanced concepts, including how to tailor your Forum environment to your special needs. Section 7 contains detailed descriptions of all the Forum requests. Section 8 is an alphabetical reference section of users' commands. Section 9 describes the duties of the chairman of the meeting. Section 10 is an alphabetical reference to the chairman's commands. Appendix A contains information needed by system administrators. Appendix B contains the subroutine descriptions. Appendix C is a glossary of Forum terms.

Manual Conventions

A few conventions and special symbols used in this manual should be introduced before you begin to explore the Multics Forum Subsystem.

Throughout the manual, "Forum" designates the Multics Forum Subsystem and the all-lowercase "forum" refers to the Multics command invoked to use this system.

Technical or other unfamiliar terms are underlined when used the first time, and are defined in the glossary in Appendix C.

File No.: 1L53, 1U53 CY74-01

Quotation marks are used to indicate the exact spelling of a request or command name, or the way a word should appear on a line typed by a user. Requests or commands that also have a short or abbreviated name appear in parentheses following the first occurrence of the name in text--for example, when introducing the "talk" (t) request. You do not type these quotation marks or parentheses.

A convention used within examples is the use of an exclamation point (!) to indicate lines you type. The exclamation point does not appear on your terminal--you do not type it, and Multics does not type it to prompt you. Exclamation points appear only in examples, and only to show which lines you type. Remember too, that every line you type must end with a carriage return (RETURN).

Text within angle brackets (<...>) is used within examples for explanatory purposes ONLY. It is not actually typed by Multics, and it should not be typed by you.

Throughout this manual, references are made to the *Multics Programmer's Reference Manual* (Order No. AG91), *Multics Commands and Active Functions* (Order No. AG92), and *Multics Subroutines and I/O Modules* (Order No. AG93) manuals. For convenience, these are referred to in text as the Programmer's Reference manual, the Commands manual, and the Subroutines manual, respectively.

Significant Changes in CY74-01

The qedx and ted requests have been changed to require that the write request ("w") be explicitly given to save changes. The editors will query if you attempt to quit with modified buffers. In addition, the quit-force request ("qf" in qedx, and "!q" in ted) can be used to quit without being queried.

To select a transaction by Person_id, the "-from Person_id" control argument must be used. New transaction specification control arguments allow you to select transactions by date and time entered: "-after" (-af), "-after_time" (-aft), "-before" (-be), "-before_time" (-bet), "-between" (-bt), "-between_time" (-btt), "-date" (-dt). The new transaction specification keyword "highest" (last_seen) has also been added. See Section 3.

The "-by_chain" control argument has been added to the list, print and write requests to sort transactions by transaction chain. See Section 3.

The write request now accepts the "-format" (-fmt) control argument which displays the transactions on specially formatted pages with headers and footers. See Section 7 for details.

Transactions may now be deleted and retrieved by their authors as well as by the chairman. See the delete and retrieve requests in Section 7.

The "exec_com" (ec) request has been added which allows you to execute an exec_com program containing Forum requests. You can now create a subsystem start_up (start_up.fmec) to be executed each time you invoke Forum. Two new control arguments to the forum command, -start_up (-su) and -no_start_up (-nsu) govern this action.

A new switch, "adjourned" (adj), allows the chairman to temporarily adjourn a meeting, preventing users from entering it. The "add_project", "add_participant", "make_public", "remove_project", "remove_participant", and "unmake_public" requests have been added for the chairman to invoke at Forum request level. They parallel the appropriate chairman's commands. See Section 7.

The forum_chairman_$delete_forum entry point has been moved to forum_$delete_forum so that any user with modify permission on the containing directory may now delete a meeting. The forum_$forum_add_name and forum_$forum_delete_name entry points have been replaced with forum_chairman_$chname_forum and forum_chairman_$chname_forum_idx respectively. The forum_limits and trans_ref_info entry points to forum_ have been replaced with real_forum_limits, real_trans_ref_info, and trans_time_info. The forum_chairman_$expunge description has been included in this update.

CONTENTS

SECTION 1

INTRODUCTION TO FORUM

MEETINGS--WITH A DIFFERENCE

The Multics Forum subsystem lets you engage in ongoing discussions about any topic in an orderly and controlled fashion.

Forum is often described as a teleconferencing system and is sometimes described as a bulletin board system. Actually, it is a bit of both. Using Forum, you can create (convene) meetings; you can specify exactly who may participate in the meetings; and you can pose questions, make assertions, offer opinions, and express points of view to the other participants.

The system parallels the structure and function of "real" meetings. Forum provides requests that allow you to enter a meeting, see who else is participating, find out what has been said since the last time you attended the meeting, and comment on any topic under discussion.

Forum, however, overcomes the restrictions of real meetings in several important ways. Forum's meetings can last for weeks or months and, no matter when you decide to "drop in," you can read every comment made since the meeting's inception. Even if you have nothing to say in the meeting, it's a great way to keep abreast of all developments.

Another difference is that by using Forum, groups of people can effectively communicate with each other even when separated by great distance in both space and time. For instance, many meetings now in use involve people located in all parts of the United States, Canada, and much of Western Europe.

BENEFITS AND TYPICAL USES OF FORUM

User communities put Forum to many interesting uses. Frequently, special interest groups establish meetings to allow all the members of a project team to effectively communicate and coordinate their activities (a sort of project management tool). Some meetings are useful places to solicit information from a wide body of users on a given subject ("Anyone know of a good article on PASCAL implementations?"). Often, a real bulletin board meeting is put up, so that people can advertise or solicit lodgings, places to get their cars fixed, and so on. Some meetings are design review meetings, in which someone proposes a detailed design and invites comments from co-workers and others. In general, meetings can be used for any purpose whatsoever.

The main benefit of Forum can be summarized in one word: information. Forum enables the effective distribution of information from the people who have the knowledge to the people who need the information and overcomes the barriers imposed by space, time, and number of participants. It is not meant to replace electronic mail, for there is no substitute for a secure and flexible system for one-to-one or one-to-few communication. It is rather designed to establish a wider community of users, sharing their knowledge and keeping them well-informed on issues of interest.

THE MEETING

Meetings can be public meetings, which are open to everyone, or private meetings, which are open only to designated participants. The creation of a public or private meeting is an option available to the chairman when the meeting is created. Section 2 describes how to determine whether you are eligible to attend a meeting. Section 9 describes how a chairman can set up public and private meetings.

Transactions

Meetings consist of a series of transactions. Entering a transaction in a Forum meeting is equivalent to speaking in a live meeting. If you are the chairman of a meeting, you first convene (create) the meeting and decide who will be eligible to attend. Then you enter the first transaction, which states the purpose of the meeting. Eligible participants can add entries (transactions)--this collection of transactions makes up the meeting. Here are some sample transactions:

```
[0001] (5 lines) Brooks.ProjA 05/18/83  0320.9 est Mon Everybody's_meeting
Subject:  Reason for this meeting
This meeting is a general meeting, open to all, and open to discussion of
any issues related to the Multics system. Issues related to the
forum subsystem itself should be discussed in the forum_problems
(forprob) meeting, and so-called public notices should be placed in the
Bulletin_board (bb) meeting.
---[0001]---
```

This is a sample initial transaction for a meeting called "Everybody's_Meeting." Forum numbered the transaction and supplied the other information shown in the first line. [0001] is its transaction number. (The chairman of a meeting always enters the first transaction in a meeting.) The rest of the first line describes how long the text of the transaction is (5 lines), who entered it (Brooks.ProjA), and the date and time it was entered (5/18/83 0320.9 est Mon). The user Brooks (on the ProjA project) typed in the subject and the text.

```
[0002] (3 lines) Davis.YAK 05/20/83 0002.3 est Wed Everybody's_Meeting
Subject: Fortran for a Beginner
Does anyone have a really good, basic, simple book about Fortran and the
basic commands?  I just want to put a little  variety into my repertoire
of languages.
---[0002]---
```

This is a sample second transaction, entered into the meeting by a user named Davis on the YAK project. A transaction that starts discussion on a new topic is entered via Forum's "talk" (t) request (described in section 5). The talk request always prompts for the subject of your transaction.

As a participant you understand that the initial transaction and statement of purpose (see [0001] above) throws the session open to discussion. From that point on, any eligible participant can bring up any related subject or idea. The initial transaction is numbered Transaction 1, and transactions are numbered sequentially by Forum as you enter them. The meeting remains open to discussion indefinitely since meetings are open-ended and have no time limit on their existence. When the chairman feels that the meeting has fulfilled its purpose or outlived its usefulness, he or she deletes it.

Transaction Chains

A reply to any transaction begins a transaction chain--a discussion concerning a previous transaction. Entering a transaction that refers to a previous transaction is equivalent to making a comment on a previous statement in a live meeting. Here is an example:

```
[0006] (1 line) Perri.YAK  05/23/81 1005.9 est Wed Everybody's_Meeting
Subject:  Re:  Fortran for a Beginner
Try the Fortran book by McKracken (or however you spell it!)
---[0006]---
```

User Perri wanted to make a suggestion regarding transaction [0002] (shown above) so he used Forum's "reply" request (described in Section 4). The reply request automatically generated the subject as shown here. This transaction chain now consists of transactions [0002] and [0006]. Transaction chains often grow quickly as other attendees interject opinions on the same subject.

SECTION 2

GETTING STARTED

This section tells you how to enter the Forum subsystem and introduces the Forum request loop. You will find out where meetings are held on your system and how to attend them.

ENTERING THE FORUM SUBSYSTEM

The Request Loop

Once you invoke Forum, everything you type is interpreted by the Forum request processor, not the Multics command processor. You will use Forum requests to accomplish your tasks. To enter the Forum subsystem, simply issue the "forum" command:

```
! forum

 forum:
```

This enters you into the request loop. The prompt "forum:" appears on your screen which is where you are to type any of the various requests in Forum's repertoire. Forum request lines must end with a carriage return, just like Multics commands. Forum processes a request by performing the desired action, and then prompts you for another request. This cycle repeats until you quit forum (by issuing the "quit" request described below).

At this point, you are not attending a meeting, but there are a number of things you can do such as list meetings and get help.

Listing Meetings

The forum request "list_meetings" (or the short form, lsm) is used to print out a list of all the meetings that you are eligible to attend:

```
forum:  ! lsm
Meetings = 3, Changed = 0

     Everybody's_Meeting    eve
     Bulletin_board         bb
     Personal_computers     pc

forum:
```

This is a list showing the long and short names of meetings that you can participate in. Since you have not attended any meetings yet, none of them are reported as changed ("Changed = 0"). (You can get the same information with the Multics command forum_list_meetings (flsm) described in Section 8.)

After you attend some meetings, the output also prints some flags for each meeting. The flags are the individual letters in the leftmost columns. They tell you what sort of rights you have relative to each meeting and what your state is for each meeting. The "p" flag means that you are already a participant of the meeting (have previously attended the meeting). The "c" flag means that the meeting has changed (which means simply that other participants of that meeting have entered transactions since the last time you attended the meeting). The "o" flag means that you are eligible only to be an observer, i.e., you can read the transactions but cannot enter any transactions. The "n" flag means that you have the notify switch turned on, that is, that you have asked to receive an automatic online notification every time a transaction is entered in the meeting. The "r" flag means that you have removed yourself from participation in the meeting. The "j" flag indicates meetings which have been adjourned by the chairman. For a complete description of how to turn these flags on and off, see the "switch_on" and "switch_off" request descriptions in Section 7.

Getting Help

Forum offers various ways to get help at your fingertips. For a list of available Forum requests, type a "?". To get a brief description of the requests, type "list_requests" (lr). To get a detailed explanation of requests and other Forum topics, type "help TOPIC" where TOPIC is the name of a request or other topic. For a list of TOPICS for which help is available, type "list_help" (lh). These requests are explained in more detail in Section 5.

LOCATING MEETINGS

Before you attend a live meeting, you must first find out where it is being held. Meetings can be stored in the central Forum directory or meetings can be stored in a private directory created by any user. Directions for finding meetings are described below.

The Central Forum Directory

The central forum directory is the main location for meetings that have been set up for public exposure at each site. The forum_dir (fd) command/active function (fully described in Section 8) prints your site's central forum directory:

```
! fd
  The central forum directory is >site>forum_dir.
  r 08:31 0.171 0
```

Thus, to attend a meeting called "fortran" in the central forum directory, you would type:

```
! forum >site>forum_dir>fortran
```

Meetings in Private Directories

Meetings can be created and stored anywhere in the Multics storage system. In order for you to locate any of these "private" meetings, one of the participants must supply you with the full pathname of the meeting. Thus, if user JohnB creates a meeting called "new_projects" in the directory >udd>ProjA>JohnB>meetings, you can attend this meeting by entering the command:

```
! forum >udd>ProjA>JohnB>meetings>new_projects
```

Forum Search Paths

A short-cut to typing the full pathname every time you want to attend a meeting is the use of the forum search paths.

There are usually two search paths in your forum search list put there automatically every time you log in. One is the central forum directory, and the other is a directory under your home directory called "meetings" (whether it exists or not).

Issue the print_search_paths (psp) command (fully described in the Commands manual) to print your forum search paths:

```
! psp forum
  forum
       >udd>[user project]>[user name]>meetings
       >site>forum_dir
  r 08:32 0.201 3
```

You can attend any meetings kept in these directories by telling Forum only the meeting's name (not the full pathname). Forum searches these directories automatically to find it.

To allow Forum to find meetings that do not reside in the central forum directory, but that you will be attending frequently, add their pathnames to the forum search list. Suppose you want to attend meetings under JohnB's home directory. Before entering Forum, type the Multics command line:

```
! asp forum >udd>ProjA>JohnB>meetings
```

to add the pathname of the directory to the forum search paths. Or, if you have already invoked the Forum subsystem, issue the request:

```
! ..asp forum >udd>ProjA>JohnB>meetings
```

to add the location of JohnB's meetings to your already-established forum search paths. Now whenever you want to attend the User_issues meeting in that directory, simply type "forum User_issues" at Multics command level, or "goto User_issues" from Forum request level. The "goto" request is simply the request that tells Forum what meeting you want to attend.

If you use one of the above methods, you must type issue the add_search_paths command every time you log in to attend JohnB's meetings. If you modify your search list in your start_up exec_com, it enables Forum to locate these meetings every time you log in. See the New Users' Introduction--Part II for a description of the start_up.ec.

Creating a Link

If you don't want to enter the full pathname of meetings and you don't wish to add entries to your forum search list, you can create links to those meetings you want to attend. Since Forum searches in a directory called "meetings" under your home directory by default, you can use this directory for establishing the links here. First create the "meetings" directory under your home directory:

```
! create_dir [hd]>meetings
```

Then go to your "meetings" directory:

```
! cwd [hd]>meetings
```

and create a link to the specific meeting with all its names. Since a meeting is kept in a "control" segment, its full name is <meeting_name>.control. You must provide the ".control" suffix when linking:

```
! link >udd>ProjA>JohnB>meetings>User_issues.control -copy_names
```

Now for you to attend the User_issues meeting, type "forum User_issues" (or "goto User_issues" if you are already in the request loop) and Forum will locate the link for the meeting.

The create_dir, change_wdir, and link commands and the home_dir active function are fully described in the Commands manual.

ATTENDING A MEETING

There are two ways to enter a meeting. One way is to give the meeting's name as an argument to the forum command when you first invoke the subsystem. For example, to enter the Everybody's_Meeting (short_name "eve"), type:

```
! forum eve
Everybody's_Meeting:  9 new, 10 last.
The meeting is public.

forum:
```

If you are already inside Forum's request loop, use the "goto" request:

```
forum:  ! goto eve
Everybody's_Meeting:  9 new, 10 last.
The meeting is public.

forum:
```

You are now attending the meeting and have become a participant in it. The "9 new, 10 last" message tells you that there are 9 transactions that you have not yet seen and that the 10th transaction is the last one entered in the meeting. The first transaction, the chairman's initial message, is not included in the count.

The message "The meeting is public" is an eligibility message, which is printed automatically when you first enter a public meeting (if the chairman of the meeting and site administrator have turned it on--see Section 9 for details). If the meeting is private, the eligibility message gives you other information instead such as, "There are 20 users and 2 projects eligible to attend".

Now that you are attending "Everybody's_Meeting," each request that you issue pertains to this meeting until you leave it and enter a different meeting with the "goto" request.

Finding Out Who Is Chairman

To find out who is chairing the meeting, type:

```
forum:  ! chairman
Brooks.ProjA

forum:
```

The "chairman" (cm) request gives you the Person_id.Project_id of the chairman of Everybody's_Meeting.

Listing Other Participants

To find out who else is participating in the meeting, use the "list_users" (lsu) request:

```
forum:  ! list_users
Users of the >udd>ProjA>forum>Everybody's_Meeting meeting.
Flags  Person          Last [0010]    Last time attended
       JohnB.ProjA          [0009]    03/22/83 1103.3 est Mon
       LeeB.ProjA           *END*     04/07/83 0202.3 est Wed

forum:
```

On the header line, "Last [0010]" means that the last transaction in the meeting is number ten, and the list shows the transaction number of the last one read for each user. "*END*" means that the indicated user has read the proceedings through to the end (through the last transaction).

The -eligible control argument to the "list_users" request tells who is eligible to participate in the meeting, rather than who currently participates. This request line tells you who else is explicitly able or unable to attend.

Check the eligible users in the "eve" meeting:

```
forum:  ! lsu -eligible
Eligible users of the >udd>ProjA>forum>Everybody's_Meeting meeting.
The meeting is public.

The following users are not eligible to participate:

     DVader

forum:
```

The "eve" meeting in the above example is a public meeting in which the chairman has not (with one exception) specified any users or projects as eligible or ineligible to participate. The following is an example of the output from the same request line issued in a private meeting:

```
forum:  ! lsu -eligible
Eligible users of the >udd>Rockers>Marley>meetings>roots meeting.
The meeting is not public.

The following users are eligible to participate:

     MGarvey        GIsaacs

The following projects are eligible to participate:

     Rockers        IJah

The following users are not eligible to participate:

     EPresley       FDomino        Checker

forum:
```

EXITING FORUM

Maybe you've had enough for now and would like to get out of Forum. Simply use the forum "quit" (or q) request:

```
forum: ! q
r 09.19 1.007 44
```

When you receive the ready message, you know you are out of Forum and back at Multics command level.

SECTION 3

READING TRANSACTIONS

This section describes how to peruse the proceedings of a meeting. The first part of this section tells you how to list transactions, i.e., how to print a list that tells you a little bit about each transaction that was entered in the meeting. The second part of this section shows you how to print transactions, i.e., how to read the contents of the transactions that you're interested in. The third part of this section introduces the write request which is used to save transactions in a separate segment. The fourth part of this section tells you about transaction specifiers, i.e., about the ways that you can tell Forum to list, print or write only certain transactions (the ones that you specify). The last part of this section tells you how to make sure that you stay aware of new transactions in the meetings that are of importance to you.

LISTING TRANSACTIONS

You can find out summary information about the proceedings of a meeting with the "list" (ls) request. It gives you a brief description of each transaction including the transaction number, its length (how many lines it contains), the date and time it was entered, the author's name, and the subject of the transaction.

Let's assume that you're still attending "Everybody's_Meeting", which you entered in Section 2. The "list" request used at this point lists all transactions in this meeting. To obtain a summary report of these transactions, type:

```
forum:  ! list
Trans#  Lines     Date     Time    Author          Subject
[0001]    (5)   05/18/81  03:20   Brooks.ProjA    Reason for this meeting
[0002]    (3)   05/20/81  03:46   Davis.YAK       Fortran for a Begin<More>
[0003]    (3)   05/21/81  06:49   Smith.ProjB     Other Meetings
  .
  .
  .

forum:
```

If you don't want to see a list of all the transactions in the meeting, you can specify only those that meet a certain criteria by using transaction specifiers. The simplest case is specifying a transaction by its transaction number. For example, typing "list 1" will print only a summary line about transaction [0001]. See the section on "Transaction Specifiers" below.

PRINTING TRANSACTIONS

Transaction [0001] states the reason for this meeting. To print it on your terminal, use the "print" (pr, p) request with the transaction specifier "1":

```
forum:  ! print 1
[0001] (5 lines) Brooks.ProjA 05/18/81  0320.9 est Mon eve
Subject:  Reason for this meeting
This meeting is a general meeting, open to all, and open to discussion
of any issues related to the Multics system. Issues related to the
forum subsystem itself should be discussed in the forum_problems
(forprob) meeting, and so-called public notices should be placed in
the Bulletin_board (bb) meeting.
---[0001]---

forum:
```

You can find out through the list request (see above) which other transactions you want to print. Use their transaction number to do so (e.g., "print 2"), or use a string of transaction numbers (e.g., "print 2 3 4"). To go through and print them one by one by use the "next" (n) keyword; instead of "print 2", type "print n". A keyword is another kind of transaction specifier (see "Transaction Specifiers" below).

WRITING TRANSACTIONS

If you want to save the transactions of a meeting in a separate segment for use later, use Forum's "write" (w) request. This request writes a copy of each transaction matching your selection criteria (or the current transaction if you don't specify any in particular) to a segment created in your working directory. You can give the segment any name you want by using the "-pathname PATH" control argument. Forum will append the suffix ".trans" to PATH automatically if you don't supply it. If you wish to create the segment in another place in the storage system (i.e., in a directory different from your current working directory), you can give a full pathname for PATH. See the New User's Introduction--Part I for a description of pathnames. If you do not give the -pathname control argument, Forum will create <meeting_name>.trans in your working directory, where <meeting_name> is the name of the current meeting.

To write the 10th transaction in the Everybody's_Meeting to a segment named "save_this.trans" in your working directory, type the following:

```
forum:  ! write 10 -pathname save_this
Writing 1 transaction to >udd>ProjA>Your_name>save_this.trans.

forum:
```

If you also want to save transactions 15 through 20 in the same file, type:

```
forum: ! write 15:20 -pn save_this
Appending 6 transactions to >udd>ProjA>Your_name>save_this.trans.

forum:
```

As you can see, you can specify a range of transactions in the current meeting, in this case "15:20" for transactions 15 through 20 inclusive. The above request line appends the transactions to save_this.trans, separating the transactions by appending a formfeed character (new page) to each. Each transaction is printed on a separate page by default, but you can specify different delimiters (if any) with either the -separator or -no_separator control arguments (see the "write" request description in Section 7). If you save additional transactions in this segment, they too are appended to the end.

The most common use of the write request is to print copies of a meeting's transactions on a line printer. An option you can choose to format the output pages is the "-format" (-fmt) control argument. The display consists of a page header containing the current date and time, the name of the meeting, and the page number, and a footer containing the subject of the first transaction on the page. The header and footer are separated from the text by a line of underscores and a blank line. Transactions are only split across pages if they are longer than one page. If a transaction does not fit on the remainder of the current page, a new page is always created. The -separator control argument may not be used when formatting output.

TRANSACTION SPECIFIERS

You can reference individual transactions or groups of transactions in a meeting by using transaction specifiers. The list, print, write, delete, retrieve, reset, reply and apply requests all use transaction specifiers which can be transaction numbers, keywords, control arguments, and qedx-type regular expressions. Each type causes Forum to search the current meeting for all transactions that meet a certain criteria spelled out by the transaction specifiers.

Transaction Numbers

Forum assigns numbers to transactions as it enters them into the proceedings of a meeting. When you enter a meeting for the first time, Forum automatically takes you to the first transaction, [0001]. You can reference any single transaction by its number (i.e., "print 3"), reference a range of transactions by putting a colon between numbers (i.e., "print 1:5"), or reference a multiple of individual transactions by numbers separated by a space (i.e., "print 3 5 9").

CURRENT TRANSACTION

Forum keeps track of the current transaction just as it keeps track of the current meeting. As you refer to other transactions, the last one you have dealt with is always the current transaction. Suppose you print transaction [0010]. When it has printed and the prompt waits for your next request, the current transaction is [0010]. When you then type "print +5", Forum prints transaction [0015], that is, current (which Forum knows without being told its number) plus 5 equals [0015]. You can use a minus sign (-) in the same manner.

Keywords

Keywords let you refer to transactions by their order in the proceedings without knowing their actual numbers. Following is a list of keywords:

all, a
refers to all transactions (same as first:last).

current, c
refers to the current transaction.

first, f
refers to the first transaction in the proceedings.

last, l
refers to the last transaction in the proceedings.

new
refers to the transactions that you have not yet seen.

next, n
refers to the transaction immediately after the current transaction.

previous, p
refers to the transaction immediately before the current transaction.

unprocessed, u
refers to the unprocessed transaction (i.e., a transaction that you have typed but have not yet entered into a meeting).

highest, last_seen
refers to the highest-numbered transaction printed or written.

CHAIN-TRACING KEYWORDS

Forum also supplies keywords that specifically deal with transactions that have been linked together through the use of the "reply" request (described in Section 4). If you are trying to follow transactions in a chain (which are often not entered immediately following one another in a meeting), use these keywords:

allref, aref
refers to all transactions in the current chain of transactions.

firstref, fref
refers to the first transaction in the current chain of transactions.

lastref, lref
refers to the last transaction in the current chain of transactions.

nextref, nref
refers to the next transaction in the current chain of transactions.

priorref, pref
refers to the previous transaction in the current chain of transactions.

restref, rref
refers to all remaining transactions in the current transaction chain, but not including the current transaction.

Some keywords can be combined with, or completely replace transaction numbers, and can be used with the simple operators "+" or "-", as in "last-4", "nref+1", and so on. A range of transactions can be given by 2 keywords separated by a colon (:) as in "first:last". See "Examples" below.

THE CHAIRMAN'S MESSAGE

The "chairman_message" (cmsg) keyword refers to the message or greeting that the chairman can set up when the meeting is created (or later, with the "set_message" request described in section 8). It is the message printed the first time a user enters a transaction in a meeting. This transaction specifier can be used with the print, write, and delete requests.

USE OF KEYWORDS AS REQUESTS

All of the keywords except for "unprocessed" and "chairman_message" can also be used as requests. They print the number of the transaction(s) that the keywords refer to. The "next" and "previous" keywords and all of the keywords that reference chains can be given an argument to use as the current transaction. The argument to these requests must be a transaction number. Thus, "aref 4" returns a list of all transaction numbers that refer to transaction [0004]:

```
forum:  ! aref 4
4 5 7 10

forum:
```

USE OF KEYWORDS AS ACTIVE REQUESTS

An active request is a request enclosed in brackets that is first expanded to the transaction number(s) it represents before the entire request line is executed. For example, "print [aref 4]" expands to "print 4 5 7 10" thus printing all those transactions referring to transaction [0004].

Transaction Specification Control Arguments

Control arguments are another type of transaction specifier that further restrict which transactions are selected by a request. A control argument is always preceded by a "-". They can be used to specify transactions that were authored by a certain person or that match a given character string, to choose transactions based on the date and time they were entered, or specify that all the transactions selected be grouped together by chain. The control arguments that are accepted by each request are listed for each request in section 8.

PERSON_IDS

The "-from Person_id" control argument indicates that only transactions authored by the person identified by Person_id should be selected. Thus, "-from LeeB" selects

all transactions entered by the participant LeeB. Person_ids cannot contain the asterisk ("*") or period (".") characters.

DATE/TIME SELECTION

The following control arguments select transactions according to the date and time that the transaction was entered. The strings DT, DT1, and DT2 are standard Multics date-time strings acceptable to the convert_date_to_binary_ subroutine (described in *Multics Subroutines and I/O Modules*, Order No. AG93). Each must be enclosed in quotes if it contains spaces. (Examples are "8/19/83 1500.0 edt Fri", "August 8", "5:00 pm 21 May 1984", and midnight.)

-after DT, -af DT
: selects all transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
: selects all transactions entered after the date-time specified.

-before DT, -be DT
: selects all transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
: selects all transactions entered before the date-time specified.

-between DT1 DT2, -bt DT1 DT2
: selects all transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
: selects all transactions entered between the date-times specified, inclusive.

-date DT, -dt DT
: selects all transactions entered on the day specified.

SORTING TRANSACTIONS BY CHAIN

The "-by_chain" control argument can be used to sort selected transactions into transaction chains if you are referring to more than one chain of transactions. This is useful when, for example, you want to write the last 50 transactions to a segment and you want those transactions to be grouped by transaction chain instead of the order they were entered into the meeting.

Regular Expressions

Simple regular expressions are character strings enclosed in slashes (/). They are used for selecting transactions which contain a match for a given character string. The control arguments "-subject" (-sj) and "-text" (-tx) can be given immediately before a regular expression to specify that matching is to be done only against the subject or the text respectively. By default, matching is done against both the subject

and the text. For example, "list /television/" lists all transactions which contain the string "television".

If the regular expression contains spaces, square brackets, or parentheses, it must be enclosed in quotes. If it contains quotes, then the quotes inside the slashes must be doubled, and the entire regular expression must be quoted. The following is a list of valid regular expressions:

```
/one/
"/one or/"
/FOR/
/:/
"/character(s),/"
"/He said, ""/"
```

The null regular expression, //, signifies the last regular expression used.

The string is taken quite literally, i.e., "/television/" will NOT match a transaction containing the string "Television". In order to ensure that case is not an issue, it is recommended that you use "/elevision/" which is still a unique enough string for matching only what you want.

The following characters have specialized meanings when used in a regular expression:

*
: signifies any number (or none) of the preceding character.

^
: when used as the first character of a regular expression, signifies the (imaginary) character preceding the first character on a line.

$
: when used as the last character of a regular expression, signifies the (imaginary) character following the last character on a line.

.
: matches any character on a line.

For more information on regular expressions, refer to the *Qedx Text Editor User's Guide*, Order No. CG40.

Examples

The examples below show the use of Forum transaction specifiers combined with the "print" (p) request. You can use any of the other Forum requests that make sense in each case.

p 14
: prints transaction 14.

p u
prints the unprocessed transaction.

p +5
prints the fifth transaction after the current transaction.

p /budget/
prints all transactions containing the string "budget".

p p:l -from JBSmith
prints all transactions in the range previous (p) through last (l) entered by participant JBSmith.

p fref
prints the first transaction in the current chain of transactions.

p new
prints all transactions not yet seen.

p 1:c-5 /pl1/ -from Baker
prints all transactions in the range 1 through the 5th before the current that contain the string "pl1" and were entered by participant Baker.

p aref /solutions/ -from JohnB
prints all transactions in the current reference chain that contain the string "solutions" and were entered by JohnB.

p "/one or/"
prints all transactions containing the string "one or".

p aref -before 8/19/83
prints all transactions in the current reference chain entered before 8/19/83.

p -after_time "2/2/83 12:00 pm"
prints all transactions entered after noon on 2/2/83.

p -between 12/1/83 12/31/83
prints all transactions entered in December 1983.

Examples of transaction specifiers as requests:

next
returns the number of the next transaction:

```
forum:  ! next
6

forum:
```

In the above example, if the current transaction is 5, when the keyword "next" is used as a request, Forum returns "6", the number of the next transaction.

previous 10
: returns the number of the transaction before transaction 10.

fref 40
: returns the number of the first transaction in the same reference chain as transaction 40.

aref [first]
: returns the numbers of all transactions that are in the same reference chain as the first transaction.

HOW TO STAY AWARE OF NEW TRANSACTIONS

Once you begin to follow discussions in a meeting, you'll want to stay aware of new transactions in that meeting. You can find out which meetings have changed by using the "list_meetings" request with the "-changed" control argument:

```
forum: ! lsm -changed
Bulletin_board (bb)
Everybody's_Meeting (eve)
Other_meetings (meetings)

forum:
```

This request line produces a list of all the meetings found via your forum search list in which new transactions have been entered since you last attended.

If you add the -count control argument to the above request line, the output also tells you the number of new transactions in each meeting.

Another way to find out if a meeting has changed is to "goto" it. As mentioned above, when you enter a meeting for the first time Forum sets the current transaction to transaction [0001], because you haven't seen any transactions yet. As you read transactions, Forum resets your current transaction to be the last one that you have seen. In other words, if you enter a meeting for the first time and print the first ten transactions, the next time you enter that meeting your current transaction is [0010]. The transaction indicator keeps track of the current transaction (highest number seen) so that it can keep you informed of new transactions. This is done automatically and is relayed to you whenever you enter a meeting:

```
forum:  ! g bb
Bulletin_board meeting:  2 new, 15 last.
The meeting is public.

forum:
```

In the above example you can see that when you enter the Bulletin_board meeting, Forum tells you "2 new, 15 last". This means that you have seen the first 13

transactions already, and that 2 new ones have been entered since then--transactions 14 and 15 are new and you haven't seen them yet. If you "list" the transactions in this meeting, you'll see that Forum prints an asterisk next to transaction 13, to indicate that 13 is your current transaction. When you enter a meeting, your current transaction is the last one you saw (highest-seen) when you last attended. You can reset your current transaction with the Forum "reset" request (see the forum command description in Section 7).

If there is a burning issue being discussed in a meeting and you must be aware every time a new transaction is entered, tell Forum that you need immediate notification. The forum_accept_notifications (fant) command (described in Section 8) informs Forum from command level that, although you may not be currently attending, you want Forum to send you a message every time a new transaction is entered in the specified meeting for the duration of this login session. To specify which meeting(s) you want to receive notifications about, go to the meeting and turn the notify flag (or switch) on with the switch_on request:

```
! fant
r 09:59 0.216 23

! forum

  forum: ! g personal_computers
  personal_computers meeting:  3 new, 5 last.

  forum: ! swn notify

  forum:
```

The "switch_on notify" request line turns your notify switch on for that meeting, until you explicitly turn it off with the "switch_off" request. You can also set the switch from outside of the meeting by adding "-meeting personal_computers" to the end of the "switch_on" request line.

With the notify flag set to on, the next time someone enters a transaction in that meeting you will receive a message of the form:

```
From Fox.ProjA (forum) 05/18/82 1450.2 edt Tue:
A new transaction has just been added to the
>udd>ProjA>mtgs>personal_computers meeting.
```

This message from Forum tells you that participant Fox entered a new transaction in the personal_computers meeting. But remember, you must issue the forum_accept_notifications command every time you log in, even though you need only issue the "switch_on notify" request once. Although the notify flag stays turned on permanently (until you request otherwise), each time you log in you must explicitly accept the notifications. The forum_accept_notifications command can be inserted in your start_up.ec.

If you need no further notifications, use the "switch_off notify" request to turn the notify flag off. If you want to leave the flag turned on but do not wish to

receive further notifications in your current login session, use the forum_refuse_notifications (frnt) command (described in section 9).

INTERRUPTING LENGTHY OUTPUT

When you ask for a list of all transactions in a meeting (or anything that fills your screen with output), occasionally you may decide that you do not want to wait for all the lines to be printed. You can interrupt processing of your request by pressing the QUIT key (sometimes labeled ATTN, BREAK, or INTERRUPT). This action takes you out of Forum and back to command level; Multics prints QUIT and a ready message containing the word "level" and a number. Type the Multics command "program_interrupt" or "pi" (described in the Commands manual) to reenter Forum. Your request has been aborted; now you are in the request loop and can issue another request. (At this point, the current meeting is still the same as before you QUIT.

```
forum:  ! list
Trans#  Lines     Date     Time    Author          Subject
[0001]    (5)   05/18/81  03:20   Brooks.ProjA    Reason for this meeting
[0002]    (3)   05/20/81  03:46   Davis.YAK       Fortran for a Begin<More>
[0003]    (3)   05/21/81  06:49   Smith.ProjB     Other Meetings
  .
  .
  .
QUIT   <USER  PRESSED THE INTERRUPT KEY HERE>
r 14:00 11.304 534 level 2
! pi
forum:
```

SECTION 4

SPEAKING

Once you get the irresistible urge to speak in a meeting, you can choose one of two Forum requests to do so:

- "talk"
  if you want to speak about a new topic.
- "reply"
  if you want to respond to an earlier transaction.

This section describes how to use these requests, how to edit your transaction before entering it into a meeting, how to delete and retrieve any transaction that you've authored.

OPENING A NEW TOPIC--THE TALK REQUEST

If you are currently attending a meeting and have something new to say, use the "talk" (t) request. Forum prompts you for the subject of the transaction:

```
forum:  ! talk
Welcome to Everybody's_Meeting
Subject:
```

The message "Welcome to Everybody's_Meeting" is the chairman's greeting message which, if set by the chairman, appears the first time you speak in a meeting. (See the "chairman_message" transaction specifier in Section 3.) After the "Subject:" prompt, type in a title and another carriage return. Now Forum responds with another prompt, indicating that you can proceed with the text of your transaction.

```
forum:  ! talk
Welcome to Everybody's_Meeting.
Subject:  ! home computers meeting
Transaction:
```

As you type in your transaction, keep in mind that the # and @ characters are always available for correcting or erasing characters in the line you are currently working on.

If you're satisfied with entering your transaction as is, without editing it, then the simplest way to conclude your transaction is to type a period alone on a line, and then a carriage return. As soon as you do this, the transaction is entered into the meeting, and you receive a confirmation of the form, "Transaction [0015] entered into Everybody's_Meeting." This is followed by the Forum prompt, indicating that you have been returned to Forum request level.

Here is an example of building and entering a transaction while attending a meeting. Note the use of the # character to correct a mistake in the message text.

```
  forum:  ! talk
  Welcome to Everybody's_Meeting.
  Subject:  ! home computers meeting
  Transaction:
! Is there sufficient interest too#
! create a meeting for home
! computers?  t#To help solve
! problems, some sort of information
! exchange might be useful.
! Any takers??
! .
  Transaction [0015] entered in >udd>ProjA>Everybody's_Meeting meeting.

forum:
```

When you end your transaction with a period, however, you miss the chance to check and edit it before it is entered into the meeting. Forum has a built-in text editor that gives you an easy way to edit your transactions before entering them (see "Editing Your Transaction" below).

COMMENTING ON AN OLD TOPIC--THE REPLY REQUEST

As we know, the pleasure of a "real" meeting is not just speaking or hearing the comments of others, but agreeing with wonderful ideas and challenging the mediocre ones. The Forum "reply" (rp) request works much the same as talk, except that your subject line contains an automatic reference to the transaction you are replying to, instead of a new subject. Assume that you have just printed transaction 15, shown above, regarding home_computers and wish to show your support:

```
  forum:  ! reply 15
  Subject: Re: home computers meeting
  Transaction:
  ! Yes, I second the motion!
  ! .
  Transaction [0016] entered in >udd>ProjA>Everybody's_Meeting meeting.

  forum:
```

The reply request assumes the subject of the reply to be a reference to the subject line of the transaction you are replying to. You are then prompted for the text of your reply ("Transaction:").

Once your reply is entered, it is linked to the transaction to which it is a reply, forming a transaction chain. Transaction chains are very easy to follow via the keywords specifically designed to follow the progress of chains (see "Chain-tracing Keywords" in Section 3).

EDITING YOUR TRANSACTION

Forum offers two built-in editors to allow you to change, delete, and add to your transaction while you remain in Forum. You can enter them by typing either "qedx" (or qx) to enter the Qedx editor or "ted" to enter the Ted editor.

The Qedx editor is introduced in the New Users' Introduction--Part I, and explained fully in the *Qedx Text Editor Users' Guide* (Order No. CG40). You are strongly encouraged to turn to one or both of those manuals, because this manual contains only a review of the simplest subset of editor requests.

Ted is a slightly more sophisticated editor than Qedx, and offers powerful requests and macro capabilities. See the *Multics Text Editor (TED) Reference Manual* (Order No. CP50) for a complete description of this editor. In the examples shown in this section, the qedx request is used, although you can use the ted request similarly to accomplish the same tasks.

When you are first typing in your transaction and you want to make changes, type \f alone on a line (instead of a period):

```
Transaction:
!   <text as shown above>
! Any takers??
! \f
```

This enters you into the Qedx editor where you can clean up your transaction.

Editor Requests--Qedx

Once you are in the editor, you issue editor requests as opposed to Forum requests. Here is a list of basic editor requests (the "$" means the last line in your transaction, regardless of its line number):

REQUEST	DESCRIPTION	EXAMPLES
p	prints the specified line(s)	p 2p 1,3p
=	prints the line number of the specified line	= $=
d	deletes the specified line(s)	d 3d 1,$d
a	adds lines of text after the specified line	a 2a
s/old/new/	substitutes every occurrence of the first character string with the second character string, on the specified line(s)	s/hte/the/ 1,$s/11:00/9:30/

`s/old/new/p`	same as above, but also prints the changed line	`s/hte/the/p`

To leave the qedx editor, type w (write) to save the changes, and then q (quit) to return to the Forum request loop. The write request may be used as often as desired to checkpoint the editing process. Note that these are qedx editor requests, not Forum requests (the Forum quit request is described under "Exiting Forum" in Section 2; the Forum write request is described under "Writing Transactions" in Section 3).

If the quit (q) request is issued and the transaction has been modified since it was last written, the editor will query for permission to exit. If permission is given, any changes made since the last write request will be lost. The quit-force (qf in qedx or q in ted) request can be used to abort unwanted editing of the transaction without being queried.

To restore the original transaction text after you have made changes but before you issued a write request, type "1,$dr". If you did issue a write request, this sequence will restore the text as it was last written.

Below is an extended example of how the transaction shown above might have been constructed. Supplemental comments are displayed to the right of the example. Spaces that would not necessarily be in an actual session are included here for clarity.

```
  forum:  ! talk
  Subject:  ! home computers meeting

  Transaction:
! Is there sufficent interest          <a first draft>
! create a meeting for home
! cmoputers?  To@
! computers? To help solve
! problms some sort of information
! exchange might be useful.
! Any takers??
! \f                                   <enter edit mode>

! ls/ce/cie/p                          <correct one spelling error>
  Is there sufficient interest

! s/est/est to/                        <add more>

! 1,$p                                 <print entire message>
  Is there sufficient interest to
  create a meeting for home

  computers?  To help solve
  problms some sort of information
  exchange might be might be useful.
  Any takers??
                                       <--
! 5                                    <print line 5>
  problms, some sort of information
! s/lms/lems/p                         <correct another error,>
  problems, some sort of information   <   and print the line >

! 3d                                   <delete empty line>
! w                                    <save the changes>
! q                                    <leave editor>

  forum:
```

You are now in possession of an unprocessed transaction. You can issue other requests (i.e., "print" to have another look at it), or further manipulate it. Maybe you want to adjust the margins or change the subject line. See the next section for more options to use on unprocessed transactions.

If you have typed in a new transaction and do not require the editor at this time, but you do not want to enter it into the proceedings of the meeting just yet, end your input with "\q" (instead of "." or "\f"). This puts you back at Forum request level where you can issue other requests to manipulate your unprocessed transaction.

```
! forum: ! talk
  Subject:  ! home computers meeting
  Transaction:
! <Text as shown above>
! \q

  forum:
```

Once you are satisfied with the contents of your transaction, issue the "enter" (en, send) request to enter it in the proceedings of the current meeting.

You can modify the contents of an unprocessed transaction in a number of ways but once you enter the transaction in a meeting, it is no longer unprocessed and you can no longer manipulate it. At request level, you can issue requests that will print, format, re-edit, and otherwise perfect the text of your unprocessed transaction.

Using the Emacs Editor

If you prefer, you can edit your unprocessed transaction with the Multics Emacs text editor, a display-oriented editor designed for use with video terminals. You can call Emacs from within Forum via the "apply" (ap) request:

```
forum:  ! apply emacs

<system enters Emacs, showing transaction in buffer>
```

This places your unprocessed transaction in an Emacs buffer where you can edit it. You must save any changes with the Emacs write request (^X^S) and quit the editor with ^X^C. Once you quit the editor, you are returned to the Forum request loop where you can enter your transaction with the "enter" (e) request.

For a complete description of the Emacs editor, see the *Emacs Text Editor Users' Guide*, Order No. CH27.

ENTERING A PRE-WRITTEN TRANSACTION

The talk and reply requests accept the "-input_file" (-if) control argument which allows you to enter a free-standing segment as the text of your transaction. Forum automatically supplies the header information, and (with the talk request) prompts you for the subject. For instance, if your comments were contained in a segment named "home_computers_talk" they could be entered in this way:

```
forum:  ! talk -if home_computers_talk
Subject:  ! home computers meeting
Use the "enter" request to enter the transaction.

forum:
```

Notice that you are now in possession of an unprocessed transaction. Forum reminds you to use the enter request to enter it into the proceedings of the meeting.

HANDLING AN UNPROCESSED TRANSACTION

If you have an unprocessed transaction and you "goto" a different meeting, Forum tells you that you left something behind:

```
forum (goto): Warning, there is an unprocessed transaction in the
              Everybody's_Meeting meeting.
```

In the above example, you built the transaction in "Everybody's_Meeting" and then attended a different meeting without first entering the transaction. The warning message reminds you that there is an unprocessed transaction to deal with--probably you'll want to enter it in "Everybody's_Meeting" from your current location ("en -meeting eve") or throw it away ("dl unproc").

You can use the "." (dot) request to check the subject and number of lines in your unprocessed transaction. For example, if you're attending the meeting in which you created the transaction, the output resembles this:

```
forum:  ! .
forum 1.8: 72 new, 277 last, 205 current.
Attending the >udd>ProjA>forum>Everybody's_Meeting meeting.
1 line unprocessed.
subject: home computers meeting

forum:
```

If you're attending a meeting other than the one in which you built the unprocessed transaction, the output tells you where that transaction is:

```
forum:  ! .
forum 1.8: 0 new, 8 last.
Attending the >udd>ProjA>mtgs>Other_Meetings meeting.
1 line unprocessed in the Everybody's_Meeting meeting.
subject: home computers meeting

forum:
```

If you attempt to exit Forum via the "quit" request while you have an unprocessed transaction, Forum warns you that you have an unprocessed transaction (which will be thrown away if you quit now) and asks you if you really want to quit before entering your transaction in a meeting.

Manipulating Your Transaction

If you print your unprocessed transaction, you will see question marks in place of a transaction number. Remember that if you don't end a transaction with a period when you first type it in, you must explicitly enter it before you exit Forum, or else Forum throws it away. Use the Forum "enter" (en, send) request to enter an unprocessed transaction.

THE PRINT REQUEST

The forum "print" (p) request displays text of the transaction preceded by a version of the transaction header that indicates an unprocessed transaction. The example transaction, which is your current transaction, is used for illustration:

```
forum:  ! p
[????] (3 lines) Your_name.Your_project **UNPROCESSED**
Subject:  home computers meeting
Is there sufficient interest to
create a meeting for home
computers?  To help solve
problems, some sort of information
exchange might be useful.
Any takers???
---[????]---

forum:
```

THE FILL REQUEST

The "fill" (fi) request reformats the text of your transaction by adjusting the right margin so that no line has more than a certain number of characters. By default, the maximum line length is set at 72 characters but if you prefer, you can specify another length with the -line_length (-ll) control argument followed by the maximum number of characters you want on a line.

```
forum::  ! fill -ll 50; p
[????] (4 lines) Your_name.Your_project **UNPROCESSED**
Subject:  home computers meeting
Is there sufficient interest to create a meeting
for home computers?  To help solve problems, some
sort of information exchange might be useful.  Any
takers???
---[????]---

forum:
```

THE SUBJECT REQUEST

The "subject" (sj) request can be used to print the subject of your unprocessed transaction:

```
forum:  ! subject
Subject:  home computers meeting
```

Suppose you decide that you want to change the subject of your transaction. The subject request can also be used to change the subject. If the new subject contains spaces, it must be quoted.

```
forum:  ! subject "Personal Computers"

forum:  ! subject
Subject:  Personal Computers
```

DELETING TRANSACTIONS

Transactions can be deleted (and retrieved) as described below.

The delete Request

You can delete any transaction that you have entered, for whatever reason, by using the Forum "delete" (dl) request, making the transaction unavailable to other participants. Only you and the meeting chairman have access to them. Only the chairman can delete a transaction authored by someone else. Your deleted transactions can be read again with the list, print and write requests if you give the -include_deleted or -only_deleted control arguments to these requests.

Deleted transactions remain deleted indefinitely but also remain accessible to you until the chairman issues the "expunge" request which kills deleted transactions permanently. (See "expunge" in Section 7.)

The retrieve Request

It is also possible, once you have deleted a transaction, to retrieve it again and thus re-enter it into the meeting. The "retrieve" (rt) request does just that. Only transactions that were deleted by you (and therefore authored by you) can be retrieved by you. Only the chairman can retrieve a transaction deleted by someone else. See the full description of the retrieve request in Section 7.

If the chairman has issued the "expunge" request since the transaction was deleted, then it cannot be retrieved.

SECTION 5

HOW TO GET HELP

Forum supplies several help requests that give you information about Forum requests in graduated levels of detail.

THE ? REQUEST

When you forget the short name of a request or need to be reminded of the names of any of the basic requests, use the "?" request. It prints a memory-jogging list of the most important requests.

```
forum:  ! ?
Available forum requests:

.                       forum_dir, fd           ted
goto, g, go             list_meetings, lsm      write, w
list, ls                list_users, lsu         help
print, p, pr            qedx, qx                list_help, lh
quit, q                 reply, rp               list_requests, lr
apply, ap               reset, rs               abbrev, ab
chairman, cm            retrieve, rt            exec_com, ec
current_meeting,        subject, sj             do
   cmtg                 switch_off, swf         if
delete, dl              switch_on, swn          answer
enter, en, send         talk, t                 execute, e
fill, fi

Type "list_requests" for a short description of the requests.

forum:
```

THE LIST_REQUESTS REQUEST

The "list_requests" (lr) request gives you a one-line description of each Forum request. It also tells you how you how you can send a command line to Multics from within Forum and how you can get more detailed help on any of the requests in this list.

```
forum: ! list_requests
Summary of forum requests:

Use "..COMMAND_LINE" to escape a command line to Multics.
Type "list_help" for a list of topics available to the help request.
Type "help TOPIC" for more information on a given topic.

.                          Print current status.
help                       Obtain detailed information on forum.
list_help, lh              List topics for which help is available.
list_requests, lr          List brief information on forum requests.
quit, q                    Exit forum.
goto, g, go                Enter a meeting.
print, p, pr               Print the specified transactions.
list, ls                   List the specified transactions.
abbrev, ab                 Turn abbreviation processing on or off.
     .
     .
     .
     .
     .

forum:
```

There are many more requests that normally appear in the list but are not displayed in this example.

THE LIST_HELP REQUEST

The "list_help" (lh) request gives you a list of every single possible Forum request, or a list of requests related to the TOPIC that you name. For example:

```
forum:  ! list_help ready
Topics available for forum:

ready_off
ready_on
ready

forum:
```

You can ask for detailed, specific help on any topic returned by list_help by naming it as the TOPIC argument to the "help" request (described below).

THE HELP REQUEST

For detailed information on how to use a particular Forum request, type "help" followed by the name of the request. (i.e., "help reply"). The "help" request offers complete information on every Forum request and on other selected topics concerning Forum.

The "help" request enters you into a sub-request loop, asking you at several points if you want more information ("More help?"). The most basic answers to this prompt is either "yes" or "no". A response of "?" gives you a list of all possible answers known to the help request. For a more detailed explanation of the help request, see section 8.

WHERE AM I?

If you've lost your place in the request loop, you can find out exactly where you are at any time with the Forum "." (dot) request. If you are not currently attending a meeting, it simply gives you Forum's version number:

```
forum: ! .
forum 1.10

forum:
```

If you are attending a meeting, it tells you the location of the meeting and transaction information in addition to Forum's version number:

```
forum:  ! .
forum 1.10:  9 new, 10 last.
Attending the >udd>ProjA>forum>Everybody's_Meeting meeting.

forum:
```

If you have seen some, but not all, of the transactions in this meeting, this request gives you the number of the last transaction you saw.

This request also tells you if you have any unprocessed transactions (unprocessed transactions are created when you speak in a meeting--they are fully described in Section 4).

SECTION 6

ADVANCED FORUM FEATURES

Many of the Forum components already discussed, such as control arguments on the command line, transaction editing, and powerful request language, have additional features that enable you to use Forum to perform more specialized functions. These advanced techniques derive their strength by using command level capabilities from within Forum.

CONTROL ARGUMENTS AND REQUESTS

Almost all Forum requests take control arguments. Here are some examples showing a specialized usage of the list_users request.

To find out if the user JPSea is a participant of the current meeting, type:

```
forum:  ! lsu -user JPSea
```

and if she is, Forum prints the summarization line showing last transaction seen, last time attended, and any flags for JPSea.

To get a listing of all users who are currently attending the meeting, type:

```
forum:  ! lsu -attending
```

ACTIVE REQUESTS

An active request is a request line enclosed in brackets that is first expanded to its return value before the entire request line is executed. The following is a list of Forum active requests:

current_meeting, cmtg	list, ls
do	list_meetings, lsm
exec_com, ec	list_users, lsu
execute, e	subject, sj
forum_dir, fd	subsystem_name
if	subsystem_version

The "cmtg" active request returns the pathname of the current meeting you are attending. Wherever you type "[cmtg]" on a request line, Forum replaces that with the name of the current meeting. One way of using this active request is:

```
forum:  ! write -pn [cmtg]
```

which places the current transaction in a saved segment with the name of the current meeting as its name.

With the "e" active request, you can incorporate Multics active functions into Forum request lines. Here are a few examples:

```
forum:  ! write 3 -pn [e home_dir]>feline
```

for saving transaction [0003] in a segment named "feline" in your home directory;

```
forum:  ! lsm -part -user [e last_message_sender]
```

for determining which meetings the user who last sent you an interactive message participates in;

```
forum:  ! pr -date [e date]
```

for printing all transactions in the current meeting that were entered today.

For a detailed discussion of the usage of each active request, refer to section 8.

ESCAPING TO COMMAND LEVEL

There are several ways to use the Multics command environment while you remain inside Forum. This ability can be handy for a variety of purposes.

The ..COMMAND_LINE Escape

To issue a Multics command within Forum, simply type two periods directly after the prompt, followed by the command line:

```
forum:  !  ..who
```

When the command has finished, you receive another Forum prompt.

You can use this escape to check on which saved transactions you have in your directory ("..list") and to attend to other Multics activities when they occur to you ("..sm Brie.ProjDog Let's eat.") without having to end your Forum session prematurely.

You are free to use any command language conventions and facilities in the same way you do outside Forum. Active functions (discussed in the New Users' Introduction--Part II) are especially useful in providing the command language with extra flexibility. Here are two examples:

```
forum:  ! ..sm [last_message_sender] Sure, I'm hungry too
forum:  ! ..eor [home_dir]>canine.mail
```

Standard quoting and semicolon conventions also apply when using the ..COMMAND_LINE escape.

Forum requests and active requests can not be used with the ".." request. The command line is made up of Multics commands and active functions only.

The execute Request

The "execute" (e) request performs a function very similar to that of the ..COMMAND_LINE escape, because it also passes the rest of the line to command level to be acted on. Before the command line reaches command level, however, it passes through the Forum request processor. This means that all forum active requests (which are enclosed in brackets as described above) get processed first. The results are placed in the request line, and then the line gets processed as a command line. To illustrate with the cmtg active request, which returns the pathname of the meeting you are currently attending, you can type:

```
forum:  ! execute list [cmtg].trans
```

which expands to:

```
forum:  execute list Everybody's_Meeting.trans
```

to see whether you have saved any transactions with the default name of the current meeting.

CONDITIONAL EXECUTION (THE IF REQUEST)

There may be times when you want to execute a request only on certain conditions, or execute a request if a condition is met and another if it is not met. The forum "if" request allows you to conditionally execute one of two request lines depending on the value of a returned active string. For example:

```
forum:  ! if [e exists seg [cmtg].trans] -then "e dl [cmtg].trans; w new"
            -else "w new"
```

which asks if a segment named <current_meeting>.trans exists in your working directory, then delete it and write all the new transactions in the current meeting to the default segment named <current_meeting>.trans. If the segment does not exist, then simply write the new transactions to the segment. The active strings being expanded here are Forum active requests. Thus, to execute a Multics active request, the "e" active request must be used. The actions to take follow the -then and -else control arguments, and they must be enclosed in quotes if they contain spaces or other command language characters. If the "-else" action is not specified and the condition returns "false", no action is taken.

The if active request returns one of two character strings to the Forum request processor depending on the value of an active string. For example,

```
forum:  ! write all [if [e equal [subsystem_version] 1.10]
            -then "-by_chain -format" -else "-fill"]
```

which will execute the request line "write all -by_chain -format" if the subsystem_version is 1.10, or "write all -fill" if it is not.

THE FORUM COMMAND

Many forum command control arguments have been described in earlier sections of this manual. They allow you to personalize your Forum environment. All of the forum command control arguments are described in the forum command description in Section 8.

To illustrate a few simple possibilities, here are some sample command lines using some control arguments that have not previously been discussed:

```
forum -meeting mgr_meeting -list -quit
forum -abbrev -brief
```

The first Forum command line above enters Forum, goes to the mgr_meeting meeting, lists all transactions, and quits, returning you directly to command level. The second example invokes Forum with abbreviation processing turned on and the briefest form of the Forum messages are to be printed.

```
forum -prompt Next -output_fill
forum -meeting pubs -no_prompt
```

The first example above enters Forum, prompts you with "Next" instead of "forum:" and fills all transactions before writing or printing them. The second example enters the pubs meeting and asks that you not be prompted at all in the request loop.

```
forum -meeting mgr_meeting -rq "write new -pn [e hd]>my; quit"
```

This command line enters the mgr_meeting meeting, writes any new transactions into a segment named my.trans (in your home directory), and exits Forum.

USING ABBREVIATIONS

If you are using one particular Forum command line frequently, you will find that creating an abbreviation for the command line, with the abbrev command, is a much simpler way of specifying that information. Here is how to use the abbrev command to make an abbreviation for one of the example command lines above:

```
! .ab my_forum forum -prompt Next -output_fill
```

Now when you want to enter forum with "Next" as your prompt, and to have your transactions filled on output, type "my_forum". In the example above, it is assumed that you have abbreviation processing turned on for your process. Refer to the New Users' Introduction--Part II for details concerning the abbrev command and the creation of abbreviations.

There is a whole set of "inverse" control arguments that are useful for reversing the effect of one (or more) of the control arguments you have included in an

abbreviation. If for example you do not always want your transactions filled on output, you can type "my_forum -no_output_fill" which is expanded into the command line:

```
forum -prompt Next -output_fill -no_output_fill
```

Whenever conflicting control arguments appear on a command line, Forum uses the last one to appear. Thus the above example restores the default filling action.

The abbrev Request

Abbreviations are a shorthand version of an otherwise lengthy or often-used request line. You must either use the -abbrev control argument when you invoke forum or use the Forum "abbrev" request to initiate abbreviation processing within the current Forum session. Forum abbreviations can either be defined at Multics command level or from within Forum via the ".." request.

You may want to define an abbreviation for the "talk" request, such as the one shown below which, upon termination of input, places your unprocessed transaction in an Emacs buffer for editing:

```
! .ab T do "talk -rql; apply emacs"
```

This abbreviation contains two request lines, thereby requiring the use of the "do" command.

FORUM EXEC_COMS

If you have specific sets of Forum requests that you use frequently, you can put them together in an exec_com segment which can be executed via the Forum "exec_com" (ec) request. The Forum exec_com segment can be written in either version 1 or version 2 exec_com (see the Commands manual) and must have the suffix of ".fmec". It is just like a Multics exec_com segment except that it contains forum request lines instead of Multics command lines. This means that to execute Multics commands from within the exec_com, you must use the "execute" request, and with Multics active functions, use the "e" active request. The following example shows the contents of a forum exec_com, and the method of executing it:

```
Contents of [home_dir]>sample_ec.fmec:

(1)    &command_line off
(2)    e cwd [e hd]
(3)    goto &1
(4)    if [e exists seg [cmtg].trans] -then "e dl [cmtg].trans"
(5)    w new -after "Sunday - 1week" -format -by_chain -bf
(6)    e eor [cmtg].trans
(7)    &quit

forum:  ! ec sample_ec eve
```

The "sample_ec.fmec" program executes the following:

(1) changes your working directory to your home_dir.

(2) goes to the meeting you specified as an argument to the exec_com request ("eve").

(3) checks to see if a segment named "Everybody's_meeting.trans" exists in your home directory and if it does, delete it.

(4) writes all new transactions that were entered after Sunday (i.e., this week's new transactions) to a segment with the default pathname of [home_dir]>Everybody's_meeting.trans. The transactions are sorted by chain, formatted for output, and no message is printed if no transactions were selected.

(5) prints the segment on the default line printer.

The forum request:

```
forum:  ! ec sample_ec ([lsm -changed])
```

executes the sample_ec exec_com for every meeting that has changed as returned by the list_meetings active request.

Forum searches the "exec_com" search list for segments with the ".fmec" suffix. An absolute pathname can also be specified which bypasses the search facility.

The Forum Start_up Exec_com

When forum is invoked, it searches for a start-up segment with the name "start_up.fmec". The start_up exec_com creates a specially tailored forum environment every time you enter forum. This segment is a Forum exec_com segment written in either version 1 or version 2 exec_com with Forum request lines. A sample start_up.fmec segment written in version 1 exec_com is shown below.

```
&command_line off
ab
lsm -changed -no_adjourned -long
&quit
```

This start_up turns on abbreviation processing and lists all meetings in which you are a participant that have changed, not including adjourned ones. If no meetings have changed, the message "No meetings have changed" will be printed instead. If you supplied the name of a meeting to attend on the forum command line, this information will be printed before entering the specified meeting and being placed in the forum request loop.

The directories that are searched for start_up.fmec are your home directory, your project directory, and >site, in that order.

The forum command allows you to bypass execution of the start_up exec_com with the "-no_start_up" control argument.

FORUM AND YOUR PROCESS START_UP

Another method of setting up your own Forum environment is to place a Forum command line at the end of your process start_up exec_com segment. For example:

```
fant; forum
```

tells Forum that from the time you log in, you want to be notified of any new transactions in any meetings where you have previously asked for immediate notification of new transactions (with the switch_on request). The rest of the command line tells Multics to enter you immediately into Forum while executing your process start_up.

SECTION 7

FORUM REQUEST DESCRIPTIONS

This section contains descriptions of the Forum requests, presented in alphabetical order. These requests are for all users except where otherwise noted. Each description contains the name of the request (including the abbreviated form, if any), discusses the purpose of the request, and shows the correct usage. Notes and examples are included when deemed necessary for clarity. The discussion below briefly describes the content of the various divisions of the request descriptions.

Name:

This heading lists the full request name and its abbreviated form.

Syntax

This part of the request description shows a single line that demonstrates the proper format to use when using (invoking) the request. Any elements in the line that follows the name of the request are explained under subsequent headings.

The following conventions apply in the syntax line:

1. Optional arguments are enclosed in braces (e.g., {path}, {User_ids}). All other arguments are required.

2. Control arguments are identified in the usage line with a leading hyphen (e.g., {-control_args}) simply as a reminder that all control arguments must be preceded by a hyphen in the actual invocation of the request.

Function

This heading discusses the purpose and function of the request and what happens when you use it.

Arguments and Control Arguments

These headings describe the other elements shown in the "Syntax" line, which tell Forum exactly what it should perform the request on, and specifically how it should be performed. Common arguments are meeting_name (the name of the meeting), STR (any character string), and N (an integer).

Control arguments are always preceded by a "-" character, and are typed as you see it (i.e., -off, -on). They sometimes have a short name as well as a long name (i.e., -brief, -bf). Some control arguments also have another argument associated with it as in "-date DT" where DT is a date-time value (i.e., -date "08/30/83 11:30 edt"). If any of these arguments contain spaces, parentheses, or quote characters, they must be enclosed in quotes.

Notes

Comments or clarifications that relate to the request as a whole are given under the "Notes" heading. Also, where applicable, the required access modes, the default operation (when invoked without arguments), and any special case information are included.

Examples

The examples show different valid invocations of the request. An exclamation mark (!) is printed at the beginning of each line typed by the user. This is done only to distinguish user-typed lines from system-typed lines. The results of each example request line are either shown or described.

Other Headings

Additional headings are used in some descriptions, particularly the more lengthy ones, to introduce specific subject matter. These additional headings may appear in place of, or in addition to, the notes.

Name: . (current)

SYNTAX

.

FUNCTION

prints information about the current version of Forum, the meeting the user is attending (if any) and the unprocessed transaction (if any). The format of the output is:

```
forum 1.8:  0 new, 220 last, 220 current.
Attending the >udd>Company>meetings>excursions meeting.
12 lines unprocessed.
subject: Re: picnic
```

If the user is attending a meeting, the number of new (unseen) transactions, the transaction number of the last transaction, and the number of the current transaction are printed. If the user has built a transaction but not yet entered it into a meeting, the subject and number of lines in that transaction are printed. If the user is about to enter an unprocessed transaction in a meeting other than the current meeting, this request also prints the name of the meeting that the transaction will be entered in (e.g., "12 lines unprocessed in the eve meeting." if attending the excursions meeting but entering a transaction in the eve meeting).

Name: ..command_line

SYNTAX

```
..command_line
```

FUNCTION

executes a Multics command line from within the Forum request loop.

ARGUMENTS

command_line
: is any Multics command line. When the command has executed, you are returned to the Forum request loop.

abbrev

Name: abbrev, ab

SYNTAX

```
ab {-control_args}
```

FUNCTION

controls abbreviation processing within Forum.

CONTROL ARGUMENTS

-off
specifies that abbreviations are not to be expanded.

-on
specifies that abbreviations should be expanded. (Default)

-profile PATH
specifies that the segment named by PATH is to be used as the profile segment; the suffix ".profile" is added to PATH if not present. The segment named by PATH must exist.

NOTES

Forum provides command line control arguments (-abbrev, -no_abbrev, -profile) to the forum command to specify the initial state of abbreviation processing.

If invoked with no arguments, this request enables abbreviation processing within Forum using the profile that was last used in this Forum invocation. If abbreviation processing was not previously enabled, the profile in use at Multics command level is used; this profile is normally [home_dir]>Person_id.profile.

See the abbrev command in the Commands manual for a description of abbreviation processing.

Name: add__participant, apt

SYNTAX

```
apt person_id {-control_arg}
```

FUNCTION

makes the person identified by Person_id eligible to participate in the current meeting. Only the chairman of the meeting can issue this request.

ARGUMENTS

person_id
is the Person_id of the participant to be added.

CONTROL ARGUMENTS

-read_only, -ro
Allows the added participant to read transactions but not to enter them.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

Name: add__project, apj

SYNTAX

```
apj project_id {-control_arg}
```

FUNCTION

Allows users from the project identified by project_id to become participants of the current meeting. Only the chairman of the meeting can issue this request.

ARGUMENTS

project_id
is the Project_id for the project to be admitted to the meeting.

CONTROL ARGUMENTS

-read_only, -ro
Allows users on the added project to read transactions but not to enter them.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

Name: answer

SYNTAX

```
answer STR {-control_args} request_line
```

FUNCTION

provides preset answers to questions asked by another request.

ARGUMENTS

STR
: is the desired answer to any question asked by request_line. If the answer is more than one word, it must be enclosed in quotes. If STR is -query, the question is passed on to the user. The -query control argument is the only one that can be used in place of STR.

request_line
: is any Forum request line. It can contain any number of separate arguments (i.e., have spaces within it) and need not be enclosed in quotes.

CONTROL ARGUMENTS

-brief, -bf
: suppresses printing (on the user's terminal) of both the question and the answer.

-call STR
: evaluates the active string STR to obtain the next answer in a sequence. The active string is constructed from Forum active requests and Multics active strings (using Forum's "execute" active request). The outermost level of brackets must be omitted (ie: "forum_list -changed") and the entire string must be enclosed in quotes if it contains request processor special characters. The return value "true" is translated to "yes", and "false" to "no". All other return values are passed as is.

-exclude STR, -ex STR
: passes on, to the user or other handler, questions whose text matches STR. If STR is surrounded by slashes (/), it is interpreted as a qedx regular expression. Otherwise, answer tests whether STR is literally contained in the text of the question. Multiple occurrences of -match and -exclude are allowed (see "Notes" below). They apply to the entire request line.

-match STR
answers only questions whose text matches STR. If STR is surrounded by slashes (/), it is interpreted as a qedx regular expression. Otherwise, answer tests whether STR is literally contained in the text of the question. Multiple occurrences of -match and -exclude are allowed (see "Notes" below). They apply to the entire request line.

-query
skips the next answer in a sequence, passing the question on to the user. The answer is read from the user_i/o I/O switch.

-then STR
supplies the next answer in a sequence.

-times N
gives the previous answer (STR, -then STR, or -query) N times only (where N is an integer).

NOTES

Answer provides preset responses to questions by establishing an on unit for the condition command_question and then executing the designated request line. If any request in the request line calls the command_query_ subroutine (described in the Subroutines manual) to ask a question, the on unit is invoked to supply the answer. The on unit is reverted when the answer request returns to Forum request level. See the Programmers' Reference manual for a discussion of the command_question condition.

If a question is asked that requires a yes or no answer and the preset answer is neither "yes" nor "no", the on unit is not invoked.

The last answer specified is issued as many times as necessary, unless followed by the -times N control argument.

The -match and -exclude control arguments are applied in the order specified. Each -match causes a given question to be answered if it matches STR; each -exclude causes it to be passed on if it matches STR. A question that has been excluded by the -exclude control argument is reconsidered if it matches a -match later in the request line. For example, the request line:

```
! answer yes -match /fortran/ -exclude /fortran_io/ -match /^fortran_io/
```

answers questions containing the string "fortran", except that it does not answer questions containing "fortran_io". It does, however, answer questions beginning with "fortran_io".

apply

Name: apply, ap

SYNTAX

```
ap {trans_specs} {-control_args} command_line
```

FUNCTION

places the text of the specified transaction into a temporary segment, appends the pathname of this segment to the end of command_line, and executes the resulting Multics command line. If an unprocessed transaction is specified, the transaction text is updated from the contents of the temporary segment after execution.

ARGUMENTS

trans_specs
: are transaction specifiers (see "List of transaction specification control arguments" below). If no trans_specs are given, the unprocessed transaction is selected. If there is no unprocessed transaction, one is created. See Section 3 for more information on transaction specifiers.

command_line
: is any Multics command line.

CONTROL ARGUMENTS

-by_chain
: specifies that transactions are to be grouped by transaction chain.

-fill, -fi
: specifies that the unprocessed transaction is to be refilled upon return from the command. This is the default unless the -no_input_fill control argument was given to the forum command or unless the -no_fill control argument was given to the talk or reply request.

-include_deleted, -idl
: processes transactions even if they have been deleted. See "Notes" below. Default is -only_non_deleted.

-initial
: processes only the transactions that are first in transaction chains.

-no_fill, -nfi
: specifies that the unprocessed transaction is not to be refilled upon return from the command. (Default)

-only_deleted, -odl
: processes only deleted transactions. See "Notes" below. Default is -only_non_deleted.

-only_non_deleted, -ondl
: processes only non-deleted transactions. (Default)

-reverse, -rv
causes transactions to be processed in the reverse order that they were given in the transaction specifier.

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
selects transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
selects transactions entered after the date_time specified.

-before DT, -be DT
selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

NOTES

The supplied command line need not be enclosed in quotes. However, if it contains parentheses, square brackets, semicolon, or quote characters they should be enclosed in quotes to prevent processing by Forum's request processor.

If more than one transaction is selected, the command line is applied to each selected transaction in turn. In order to select a deleted transaction, the user must either be the chairman of the meeting or the author of the deleted transaction.

EXAMPLES

This request can be used to edit the transaction with an editor other than qedx or ted. For example, the request:

```
! apply emacs
```

will invoke the emacs editor on the transaction.

The request:

```
! apply aref send_mail Smith -sj Transactions -if
```

will send each transaction in the current chain as mail to the user Smith.

Name: chairman, cm

SYNTAX

```
cm {meeting} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[cm {meeting}]
```

FUNCTION

prints or returns the User_id (Person_id.Project_id) of the chairman of the specified meeting, or changes the chairman of the current meeting. If no arguments are given, the name of the chairman of the current meeting is printed.

ARGUMENTS

meeting
: is the name or pathname of a Forum meeting.

CONTROL ARGUMENTS

-force, -fc
: tells Forum not to ask if the chairman should be changed when using the -set control argument.

-set new_chairman
: changes the chairman of the current meeting. new_chairman must be of the form Person_id.Project_id. Only the current chairman is allowed to change the chairman.

Name: current_meeting, cmtg

SYNTAX

```
cmtg {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[cmtg {-control_args}]
```

FUNCTION

prints or returns the entry name or pathname of the meeting that the user is currently attending.

CONTROL ARGUMENTS

-absolute_pathname, -absp
prints or returns the full pathname of the meeting.

-entry, -et
prints or returns only the entry name portion of the pathname of the current meeting. (Default)

Name: delete, dl

SYNTAX

```
dl trans_specs
```

FUNCTION

Allows participants to logically remove specified transactions from a meeting. The chairman of a meeting can delete any transaction, and a participant can delete any transaction that he/she submitted. These transactions can then be physically removed from the meeting by the chairman with the "expunge" request.

ARGUMENTS

trans_specs
are transaction specifiers that determine which transactions are deleted (see below). See Section 3 for complete information on transaction specifiers. At least one transaction must be specified with this request.

delete

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
selects transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
selects transactions entered after the date_time specified.

-before DT, -be DT
selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

NOTES

Deleted transactions are ignored by all requests that process transactions. Deleted transactions can be restored by using the "retrieve" request.

Name: delete_participant, dlpt

SYNTAX

```
dlpt Person_ids {-control_args}
```

FUNCTION

logically removes the record of participation (attendee record) of a participant in a meeting. Once removed, these attendee records can be deleted with the "expunge" request. Only the chairman of the meeting can issue either request.

ARGUMENTS

Person_ids
are the name(s) of the participants whose records are to be deleted.

CONTROL ARGUMENTS

-brief, -bf
suppresses the message "Specified switch was not changed."

-long, -lg
prints the message "Specified switch was not changed." if the record for that participant has already been deleted. (Default)

NOTES

Deleted participants are ignored by all requests that list participants. Deleted participants can be restored by using the "retrieve_participant" request.

Name: do

SYNTAX

```
     do request_string {args}
or:  do -control_args
```

SYNTAX AS AN ACTIVE REQUEST

```
[do "request_string" args]
```

FUNCTION

expands a request line by substituting the supplied arguments into the line before execution. As an active request, returns the expanded request_string rather than executing it.

do

ARGUMENTS

request_string
is a Forum request line enclosed in quotes.

args
are character string arguments that replace parameters in request_string.

CONTROL ARGUMENTS
These control arguments set the mode of operation of the do request:

-absentee
establishes an any_other handler that catches all conditions and aborts execution of the request line without aborting the process.

-brief, -bf
does not print the expanded request line before execution. (Default)

-go
passes on the expanded request line for execution. (Default)

-interactive
does not establish the any_other handler. (Default)

-long, -lg
prints the expanded request line before execution.

-nogo
does not pass on the expanded request line for execution.

LIST OF PARAMETERS

Any sequence beginning with & in the request line is expanded by the do request using the arguments given on the request line.

&1 ... &9	are replaced by the first through 9th arguments respectively.
&(N)	is replaced by argN where N is any integer.
&q1 ... &q9	is replaced by the first through 9th arguments with any quotes doubled.
&q(N)	is replaced by argN with any quotes doubled. N can be any integer.
&rf1 ... &rf9	is replaced by the first through 9th arguments surrounded by a level of quotes with any contained quotes doubled.
&r(N)	is replaced by a requoted argN. I can be any integer.
&fN	is replaced by all the arguments starting with argN where N must be a digit from 1 to 9.

&f(N)	is replaced by all the arguments starting with argN where N can be any integer.
&qfN	is replaced by all the arguments starting with argN with any quotes doubled. N must be a digit from 1 to 9.
&qf(N)	is replaced by all the arguments starting with argN with quotes doubled where N can be any integer.
&rN	is replaced by all the arguments starting with argN. Each argument is placed in a level of quotes with contained quotes doubled. N must be a digit from 1 to 9.
&rf(N)	is replaced by all the arguments starting with argN, requoted. N can be any value.
&&	is replaced by an ampersand.
&!	is replaced by a 15 character unique string. The string used is the same everywhere & appears in the request line.
&n	is replaced by the actual number of arguments supplied.
&f&n	is replaced by the last argument supplied.

Name: enter, en, send

SYNTAX

```
en {-control_args}
```

FUNCTION

enters an unprocessed transaction, that was created either by escaping to the Forum request loop via "talk" mode or by using the "qedx" or "ted" requests.

CONTROL ARGUMENTS

-brief, -bf
    suppresses the message indicating the transaction was entered.

-long, -lg
    prints a message indicating that the transaction was successfully entered. (Default)

-meeting meeting_name, -mtg meeting_name
enters the transaction into the proceedings of the meeting_name meeting. The default is to enter the transaction into the meeting that the user was attending when the transaction was created. Meeting_name can be the name or pathname of a meeting. This control argument cannot be used if the transaction was built using the "reply" request.

Name: exec_com, ec

SYNTAX

```
ec ec_path {ec_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[ec ec_path {ec_args}]
```

FUNCTION

executes a exec_com segment containing forum requests. As an active request, it returns a value passed from the exec_com segment with the &return statement.

ARGUMENTS

ec_paths
is the pathname of an exec_com program. The ".fmec" suffix is assumed if not specified.

ec_args
are optional arguments to the exec_com program and are substituted for parameter references in the program.

NOTES

Forum searches the "exec_com" search list for ec_paths with the ".fmec" suffix unless an absolute pathname is given.

For a description of the exec_com language (both version 1 and version 2), see the Commands manual.

When evaluating a subsystem exec_com program, Forum active requests are used rather than Multics active functions when evaluating the &[...] construct and the active string in an &if statement. The subsystem's execute active request must be used to evaluate Multics active strings within the exec_com.

Name: execute, e

SYNTAX

```
e LINE
```

SYNTAX AS AN ACTIVE REQUEST

```
[e LINE]
```

FUNCTION

executes the supplied line as a Multics command line. As an active request, evaluates a Multics active string and returns the result to the Forum request processor.

ARGUMENTS

LINE
is the Multics command line to be executed or the Multics active string to be evaluated. It need not be enclosed in quotes.

NOTES

The recommended method to execute a Multics command line from within Forum is the ".." escape sequence which does not expand Forum active requests in the line before passing it to the command processor. The "execute" request is intended as a means of passing information from Forum to the Multics command processor.

All unquoted (), [], and " characters in the given line are processed by the Forum request processor and not the Multics command processor. This fact permits the passing of the values of Forum active requests to Multics commands when using the "execute" request or, when using the "execute" active request, to Multics active functions for further manipulation before returning the values to the Forum request processor for use within a request line.

Name: expunge

SYNTAX

```
expunge {-control_args}
```

FUNCTION

physically removes deleted participant records and transactions from a meeting and frees their storage. This request also checks the consistency of the control segment and attempts to salvage invalid transaction and attendee records (see "Notes" below). Only the chairman of the meeting can issue this request.

CONTROL ARGUMENTS

-brief, -bf
suppresses the messages indicating how many participants and transactions were expunged.

-force, -fc
does not ask if the user really wants to expunge the meeting.

-long, -lg
prints messages indicating how many participants and transactions were expunged. (Default)

-participants, -part
expunges only attendee records. The default expunges both attendee records and transactions.

-transactions, -trans
expunges only transactions. The default expunges both attendee records and transactions.

NOTES

A user's attendee record is his record of participation in the meeting, which displays his User_id, last time attended, and last transaction seen. Attendee records are printed by the "list_users" request.

Name: fill, fi

SYNTAX

```
fi {-control_args}
```

FUNCTION

reformats the text of the unprocessed transaction to fit within a given line length. It is also used to control whether or not the transaction is filled after it is entered.

CONTROL ARGUMENTS

-line_length N, -ll N
is the width to be used when reformatting the text. (Default-- either the value specified by the -line_length argument to the forum command or 72 if this argument was not given). The line length given must be between 10 and 136.

-off
does not fill this transaction by default when printed or written.

-on
fills this transaction by default when printed or written.

Name: forum_dir, fd

SYNTAX

```
fd
```

SYNTAX AS AN ACTIVE REQUEST

```
[fd]
```

FUNCTION

prints/returns the absolute pathname of the central forum directory.

NOTES

The central forum directory is the location for site-approved public meetings. This directory is included in the default forum search path. The forum request:

```
! e add_search_paths forum [forum_dir]
```

can be used to add the central directory to the forum search path. See Section 3 for more information on search paths.

Name: goto, go, g

SYNTAX

```
g {meeting_name} {-control_arg}
```

FUNCTION

enters a meeting or switches from one meeting to another. This request requires access to the meeting_name meeting.

ARGUMENTS

meeting_name
is the pathname or entryname of an established meeting. If a pathname is specified, it identifies the meeting directly. Otherwise, the forum search list will be used in an attempt to locate the meeting. The ".control" suffix is optional. Either this argument or the "-meeting" control argument must be given.

CONTROL ARGUMENTS

-meeting meeting_name, -mtg meeting_name
enters the meeting whose pathname or entryname is meeting_name. See the description of the meeting_name argument above.

Name: help

SYNTAX

```
help {topics} {-control_args}
```

FUNCTION

prints information about various Forum topics including detailed descriptions of Forum requests.

ARGUMENTS

topics
are the topics on which information is to be printed. The topics available within Forum can be determined by using the "list_help" request (see "list_help" below).

CONTROL ARGUMENTS

The following specify information selection. All but -all and -brief leave you in help's request loop after they have printed their specific information.

-all, -a
prints the entire info without intervening questions.

-brief, -bf
prints only a summary of a request or active request, including the syntax section, list of arguments, control arguments, etc.

-brief_header, -bfhe
prints a brief heading with the info.

-control_arg STR, -ca STR
prints the descriptions of control (or other) arguments containing the string STR.

-header, -he
prints only the heading line.

-search STRs, -srh STRs
begins printing with the paragraph containing all the strings STRs. By default, printing begins at the top of the information.

help

-section STRs, -scn STRs
begins printing at the section whose title contains all the strings STRs. By default, printing begins at the top of the information.

-title
prints section titles and section line counts, then asks if the user wants to see the first paragraph of information.

LIST OF RESPONSES

?
prints the list of responses allowed to help queries.

.
identifies the current Forum environment.

.. command_line
treats the remainder of the response as a Multics command line.

brief, bf
prints a brief summary, including the Syntax section and a list of arguments, then repeats the previous question.

control_arg STR, ca STR
prints the descriptions of control (or other) arguments whose names contain STR.

header, he
prints a long heading line consisting of the pathname of the info seg, heading, and line count.

no, n
stops printing information for this topic and proceeds to the next topic, if any.

quit, q
stops printing information for this topic and returns to the Forum's request level.

rest {-scn}, r {-scn}
prints remaining information for this topic without intervening questions. If -section or -scn is given, "help" prints only the rest of the current section without questions and then asks if the user wants to see the next section.

search {STRs} {-top}, srh {STRs} {-top}
skips to the next paragraph containing all the strings STRs. If -top or -t is given, searching starts at top of the information. If STRs are omitted, "help" uses the STRs from the previous search response or the -search control argument.

section {STRs} {-top}, scn {STRs} {-top}
skips to the next section whose title contains all the strings STRs. If -top or -t is given, title searching starts at the top of the information. If STRs are omitted, "help" uses the STRs from the previous section response or the -section control argument.

skip {-scn|-seen}, s {-scn|-seen}
: skips to the next paragraph. If -section or -scn is given, skips all paragraphs of the current section. If -seen is given, skips to the next paragraph that the user has not seen. Only one control argument is allowed in each skip response.

title {-top}
: lists titles and line counts of the sections that follow; if -top or -t is given, "help" lists all section titles. "help" then repeats the previous question after titles are printed.

top, t
: skips to the beginning of the info seg, prints the heading line, and asks whether the user wants to see the first section.

yes, y
: prints the next paragraph of information on this topic.

NOTES

If given no topic names, the "help" request explains what requests are available to obtain information about Forum.

For a more complete description of the control arguments and responses accepted by this request, see the help command in the Commands manual.

Name: if

SYNTAX

```
if EXPR -then LINE1 {-else LINE2}
```

SYNTAX AS AN ACTIVE REQUEST

```
[if EXPR -then STR1 {-else STR2}]
```

FUNCTION

conditionally executes one of two request lines depending on the value of an active string. As an active request, returns one of two character strings to the Forum request processor depending on the value of an active string.

ARGUMENTS

EXPR
: is the active string that must evaluate to either "true" or "false". The active string is constructed from Forum active requests and Multics active strings (using the Forum's execute active request).

LINE1
is the Forum request line to execute if EXPR evaluates to "true". If the request line contains any request processor characters, it must be enclosed in quotes.

STR1
is returned as the value of the if active request if the EXPR evaluates to "true".

LINE2
is the Forum request line to execute if EXPR evaluates to "false". If omitted and EXPR is "false", no additional request line is executed. If the request line contains any request processor characters, it must be enclosed in quotes.

STR2
is returned as the value of the if active request if the EXPR evaluates to "false". If omitted and the EXPR is "false", a null string is returned.

Name: list, ls

SYNTAX

```
ls {trans_specs} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[ls {trans_specs} {-control_args}]
```

FUNCTION

prints the transaction number, author, subject, and date/time of specified transactions on the user's terminal. As an active request, returns a list of numbers of selected transactions.

ARGUMENTS

trans_specs
are transaction specifiers that determine the transactions to be listed. See "List of transaction specification control arguments" below. See Section 3 for complete information on transaction specifiers. If no transaction specifiers are supplied, all transactions are listed.

CONTROL ARGUMENTS

-brief, -bf
prints a brief summary line for each transaction selected.

-by_chain
specifies that transactions are to be grouped by transaction chain.

list

-fill, -fi
fills listed transactions in order to obtain the line count. Not valid if invoked as an active request.

-header, -he
prints the header. (Default)

-include_deleted, -idl
lists transactions even if they have been deleted. See "Notes" below. Default is -only_non_deleted.

-inhibit_error, -ihe
does not print an error if no transactions are selected. Only valid if invoked as an active request.

-initial
lists only the transactions that are at the beginning of transaction chains.

-meeting meeting_name, -mtg meeting_name
selects the specified transactions from the meeting_name meeting. The default selects transactions from the current meeting.

-no_fill, -nfi
suppresses filling of transactions in order to obtain the line count. Not valid if invoked as an active request. (Default)

-no_header, -nhe
suppresses printing of the header. Not valid if invoked as an active request.

-no_inhibit_error, -nihe
prints an error if no transactions are selected. Only valid if invoked as an active request. (Default)

-no_update, -nud
causes the list request to not update the current transaction index. Not valid if invoked as an active request.

-only_deleted, -odl
lists only deleted transactions. See "Notes" below. Default is -only_non_deleted.

-only_non_deleted, -ondl
lists only non-deleted transactions. (Default)

-output_file PATH, -of PATH
directs the output of the list request to the file named by PATH. Not valid if invoked as an active request.

-reverse, -rv
lists transactions in the reverse order of that specified.

-update, -ud
sets the current transaction index to the index of the first transaction listed unless the current transaction is among those listed. Not valid if invoked as active request. (Default)

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
selects transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
selects transactions entered after the date_time specified.

-before DT, -be DT
selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

NOTES

In order to list a deleted transaction, the user must either be the chairman of the meeting or the author of the deleted transaction.

Name: list_help, lh

SYNTAX

```
lh {topics}
```

FUNCTION

displays the names of all Forum info (help) segments pertaining to a given set of topics.

ARGUMENTS

topics
specifies the topics of interest. Any Forum info segment that contains one of these topics is listed.

NOTES

If no topics are given, all info segments available for Forum are listed.

When matching topics with info segment names, an info segment name is considered to match a topic only if that topic is at the beginning or end of a word within the segment name. Words in info segment names are bounded by the beginning and end of the segment name and by the characters period (.), hyphen (-), underscore (_), and dollar sign ($). The ".info" suffix is not considered when matching topics.

EXAMPLES

```
! list_help ex
```

matches info segments named expunge, execute, trans_specs_ex, and exec_com but does not match an info segment named nextref.

Name: list_meetings, lsm

SYNTAX

```
lsm {meeting_names} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[lsm {-control_args}]
```

FUNCTION

prints a list of selected meetings on the user's terminal. For each meeting selected, information about the names of the meeting and per-user meeting attributes are listed.

ARGUMENTS

meeting_names
are optional meeting_names. If any are supplied, information about only the specified meeting_names is printed. The star convention is supported for meeting_names. If no meeting_names are given, information for all meetings found in the forum search path are printed.

CONTROL ARGUMENTS

-absolute_pathname, -absp
prints the absolute pathname of a meeting. The default lists the long and short meeting names only.

-adjourned, -adj
selects information about meetings that have been adjourned.

-all, -a
prints information about all meetings. The default prints information about the meetings in which the user is eligible to participate only.

-brief, -bf
suppresses the message "No meetings have changed", which is printed by default if the -changes control argument is used and no meetings have changed.

-chairman {Person_id}, -cm {Person_id}
prints information about meetings of which Person_id is chairman. If Person_id is not given, the user's Person_id is used.

-changed, -chg
prints information about meetings in which the user is a participant and in which new transactions have been entered.

-count, -ct
prints out the number of new transactions for a meeting in which the user is participating. This control argument cannot be used if list_meetings is invoked as an active request.

-eligible, -elig
prints information about all meetings in which the user is eligible to participate. (Default)

-exclude meeting_names, -ex meeting_names
excludes the meetings identified by meeting_names from the output list. The meeting_names argument can be a starname. The default (-include) is to select all meetings that match the specified starname.

-from DT, -fm DT
selects meetings that have changed since the specified time. DT is any string acceptable to the convert_date_to_binary_ subroutine. The default is the current date-time.

list_meetings

-header, -he
prints the header. This is the default, unless the -changes control argument is given.

-include meeting_names, -incl meeting_names
includes the meetings identified by meeting_names in the output list. The meeting_names argument can be a starname. (Default)

-inhibit_error, -ihe
does not print warning messages for such things as bad meeting format and errors encountered while searching for meetings. (Default)

-long, -lg
prints the message "No meetings have changed" if the -changes control argument is used and no meetings in which the user is a participant have changed. (Default)

-no_adjourned, -nadj
specifies that adjourned meetings are not to be listed.

-no_changes, -nchg
does not list changed meetings.

-no_header, -nhe
suppresses printing of the header.

-no_inhibit_error, -nihe
prints all warning messages.

-no_notify, -nnt
does not list meetings in which the user has the notify flag set.

-no_participating, -npart
does not list meetings in which the user is a participant.

-no_read_only, -nro
does not list meetings to which the user has only read access.

-notify, -nt
lists only meetings in which the user has the notify flag set.

-participating, -part
lists only meetings in which the user is participating. The default prints information about all meetings in which the user is eligible to participate.

-read_only, -ro
lists just meetings to which the user has only read access.

-user User_id
determines participation, changes, and eligibility attributes for the user specfied by User_id. User_id is in the form Person_id.Project_id.tag, where any of the components can be the character "*". If a component is omitted, it is assumed to be "*".

-verbose, -vb
prints the chairman's User_id and current and last transactions for each meeting. Cannot be used with -changes or when lsm is being invoked as an active request.

NOTES ON FLAGS

The output from this command can include flags that have the following interpretation:

e eligible flag: indicates that the user can participate in the meeting.

p participant flag: indicates that the user is a participant in the meeting (i.e., has "gone to" the meeting at least once).

r removed flag: indicates that the user has removed himself from participation in the meeting.

n notify flag: indicates that the user has turned on the notify flag in the meeting (i.e., that the user has requested online notification when new transactions are entered in the meeting).

c change flag: indicates that new transactions have been entered in the proceedings of the meeting since the user last attended this meeting.

o observer flag: indicates that the user has only read access to the meeting.

j adjourned flag: indicates that the chairman has temporarily adjourned the meeting.

Note that the flags corresponding to the selection criteria are not shown (i.e., if the user is selecting meetings to which he or she is eligible, the "e" flag is not printed).

list_requests

Name: list_requests, lr

SYNTAX

```
lr {STRs} {-control_args}
```

FUNCTION

prints a brief description of selected Forum requests.

ARGUMENTS

STRs
specifies the requests to be listed. Any request with a name containing one of these strings is listed unless -exact is used, in which case the request name must exactly match one of these strings.

CONTROL ARGUMENTS

-all, -a
includes undocumented and unimplemented requests in the list of requests eligible for matching the STR arguments.

-exact
lists only those requests whose names exactly match one of the STR arguments.

NOTES

If no STRs are given, all requests are listed.

When matching STRs with request names, a request name is considered to match a STR only if that STR is at the beginning or end of a word within the request name. Words in request names are bounded by the beginning and end of the request name and by the characters period (.), hyphen (-), underscore (_), and dollar sign ($).

EXAMPLES

```
! list_requests ex
```

matches requests named execute and exec_com but does not match a request named nextref. You have to specify -all to get the expunge request listed, unless you are the chairman of the meeting you are currently attending.

Name: list_users, lsu

SYNTAX

```
lsu {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[lsu {-control_args}]
```

FUNCTION

prints a list of selected participants of a meeting. For each participant selected, the Person_id, Project_id, current transaction number (last one seen), the date/time last attended, and several flags are listed. The active request returns a list of Person_ids.

CONTROL ARGUMENTS

-all, -a
lists all participants, including those who have been "removed" from the meeting.

-attending, -at
lists only participants who are currently attending this meeting.

-brief, -bf
suppresses the message "No participants were selected."

-eligible, -elig
prints a list of users and projects that are eligible to attend the meeting. Read-only eligibility is denoted by an asterisk (*).

-header, -he
prints the header. (Default)

-long, -lg
prints the message "No participants were selected." if this is the case. (Default)

-meeting meeting_name, -mtg meeting_name
lists participants of the meeting_name meeting. The default lists participants of the current meeting.

-no_header, -nhe
suppresses printing of the name_list header.

-no_read_only, -nro
does not list read-only participants.

-notify, -nt
lists only participants who have the notify flag set.

-project Project_ids, -pj Project_ids
lists only information about participants on the specified project. All arguments following -project until the next control argument are taken as Project_ids.

-read_only, -ro
lists just the read-only participants.

-seen transaction_number
lists only the participants who have read the specified transaction.

-sort TYPE
sorts the output by TYPE. TYPE is either "name" for sorting by the Person_id of participants, or "date_time_attended" (dta) for sorting by the time the participant last attended the meeting. This control argument cannot be used if list_users is invoked as an active request.

-totals, -tt
prints only the total number of participants selected.

-unseen transaction_number
lists only participants who have not read the specified transaction.

-user Person_ids
lists only information about the named participants. All arguments following -user until the next control argument are taken as Person_ids.

NOTES

The displayed flags have the following meanings: "r" indicates that the participant has been removed from the meeting and is therefore no longer a participant, "n" means that the user has the "notify flag" turned on, and "o" means that the user is an observer and cannot enter transactions.

Name: make_public, mp

SYNTAX

```
mp {-control_arg}
```

FUNCTION

opens the current meeting to public participation. Only the chairman of the meeting can issue this request.

CONTROL ARGUMENTS

-read_only, -ro
Allows the public to read transactions but not to enter them.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

Name: print, pr, p

SYNTAX

```
p {trans_specs} {-control_args}
```

FUNCTION

prints specified transactions on the terminal.

ARGUMENTS

trans_specs
: are transaction specifiers that determine the transactions to be printed. See "List of transaction specification control arguments" below. See Section 3 for complete information on transaction specifiers. If no transaction specifiers are supplied, the unprocessed transaction is printed (if there is one), otherwise the current transaction is printed.

CONTROL ARGUMENTS

-by_chain
: specifies that transactions are to be grouped by transaction chain.

-fill, -fi
: fills transactions before printing them.

-include_deleted, -idl
: selects transactions even if they have been deleted. See "Notes" below. Default is -only_non_deleted.

-initial
: selects only transactions that begin transaction chains.

-meeting meeting_name, -mtg meeting_name
: selects the specified transactions from the meeting_name meeting. The default selects transactions from the current meeting.

-no_fill, -nfi
: suppresses filling of transactions. (Default)

-only_deleted, -odl
: selects only deleted transactions. See "Notes" below. Default is -only_non_deleted.

-only_non_deleted, -ondl
selects only non-deleted transactions. (Default)

-reverse, -rv
prints transactions in the reverse order of that given in the transaction specifier.

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
selects transactions entered on or after the date specified. The time of day is ignored.

-after_time DT, -aft DT
selects transactions entered after the date_time specified.

-before DT, -be DT
selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

NOTES

In order to print a deleted transaction, the user must be either the chairman of the meeting or the author of the deleted transaction.

Name: qedx, qx

SYNTAX

```
qx {-control_arg}
```

FUNCTION

invokes the Qedx editor to edit the text of the unprocessed transaction or to build a new transaction. See the *Qedx Text Editor User's Guide*, Order No. CG40, for more information on the Qedx editor.

CONTROL ARGUMENTS

-fill, -fi
: specifies that the transaction is to be refilled upon exit from the editor. This is the default unless the -no_input_fill control argument was given to the forum command or unless the -no_fill control argument was given to the talk or reply request.

-no_fill, -nfi
: specifies that the transaction is not to be refilled upon exit from the editor. (Default)

NOTES

The write (w) request must be used to reflect any changes made to the transaction back to forum. The write request can be used as often as desired to checkpoint the editing process.

If the quit (q) request is issued and the transaction has been modified since it was last written, qedx will query for permission to exit. If permission is given, any changes made since the last write request will be lost. The quit-force (qf) request can be used to abort unwanted editing of the transaction without being queried.

If issued before use of the write request, the request line:

```
!   1,$dr
```

will restore the original transaction text to the buffer. If given after a write request, this request line will restore the transaction as saved by the last write request given in the buffer.

Name: quit, q

SYNTAX

```
q {-control_arg}
```

FUNCTION

exits the Forum subsystem.

CONTROL ARGUMENTS

-force, -fc
does not ask if the user really wants to quit if there is an unprocessed transaction.

Name: remove_participant, rpt

SYNTAX

```
rpt Person_id
```

FUNCTION

makes the person identified by Person_id ineligible to participate in the current meeting. Only the chairman of the meeting can use this request.

ARGUMENTS

Person_id
is the Person_id of the user to be removed from participation in the meeting.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

Name: remove_project, rpj

SYNTAX

```
rpj Project_id
```

FUNCTION

makes all participants who are currently eligible because of their Project_id, not because of their Person_id, ineligible to attend the current meeting. Only the chairman of the meeting may use this request.

ARGUMENTS

Project_id
: is the Project_id of the project whose members are to be removed from participation if they lack explicit Person_id participation rights.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

Name: reply, rp

SYNTAX

```
rp {transaction_specifier} {-control_args}
```

FUNCTION

allows a participant to build, and optionally enter into the proceedings of a meeting, a transaction that is in reply to an earlier transaction and thus perpetuate a "chain" of transactions.

ARGUMENTS

transaction_specifier
: is the transaction specifier of the transaction to be replied to. If this argument is not given, the current transaction is replied to. Transaction specifier must name only one transaction and therefore cannot include a User_id, regular expression, or range.

CONTROL ARGUMENTS

-brief, -bf
: suppresses printing of the message indicating that the transaction was successfully entered.

-fill, -fi
fills the transaction after the user exits input mode. This is the default, unless the -no_input_fill (-nif) control argument was given with the invocation of forum. The default fill width is 72 but the user can specify the fill width via the -line_length NN (-ll NN) control argument to the forum command.

-input_file pathname, -if pathname
enters the segment identified by pathname into the proceedings. The default takes the transaction as it is typed in from the terminal and enters the request loop upon termination of input.

-long, -lg
prints a message indicating that the transaction was successfully entered. (Default)

-meeting meeting_name, -mtg meeting_name
enters the transaction into the proceedings of the meeting_name meeting. The default enters the transaction into the proceedings of the meeting the user is currently attending. The meeting_name argument can be a meeting name or a pathname.

-no_fill, -nfi
does not fill the transaction to the default fill width when entered. The default fills the transaction, unless the -no_input_fill (-nif) control argument was specified when forum was invoked. The fill width defaults to 72 unless the -line_length NN (-ll NN) control argument was specified when forum was invoked. (Default)

-no_request_loop, -nrql
enters the transaction into the meeting without first entering the forum request loop. (Default except when -input_file is used.)

-request_loop, -rql
enters the forum request loop before the transaction is entered in the meeting.

-subject subject_string, -sj subject_string
uses subject_string as the subject, avoiding the prompt for subject. The default prompts the user for the subject. If subject_string contains spaces, it must be quoted.

-terminal_input, -ti
inputs the transaction from the user's terminal. (Default)

TERMINAL INPUT

Unless the user makes use of the -input_file control argument, the transaction is built by taking lines from the user's terminal. There are three methods of exiting from terminal input mode:

. Enters the transaction as is, returning to Forum request level.

\f Exits terminal input mode and invokes the qedx editor on the transaction.

\q Exits terminal input mode and enters the Forum request loop.

NOTES

A transaction that was not entered via the period (.) to exit from terminal input mode can be entered from Forum request level via the "enter" request. See the description of the "enter" request for details.

Name: reset, rs

SYNTAX

```
rs {trans_specs} {-control_arg}
```

FUNCTION

resets the user's "current" or "highest transaction seen" index to the specified transaction number.

ARGUMENTS

trans_specs
is a transaction specifier that determines the value of the transaction number that the index is reset to. This can refer to only one transaction. If not specified, the default is the current transaction number. See Section 3 for complete information on transaction specifiers.

CONTROL ARGUMENTS

-current
adjusts the "current" index. (Default)

-highest
adjusts the "highest transaction seen" index.

-new
makes the specified transaction the first "new" transaction. This is equivalent to "reset -highest trans_num-1".

-next
makes the specified transaction the "next" transaction. This is the same as "reset trans_num-1".

NOTES

The "current transaction index" is a logical pointer to a transaction in a meeting. Many Forum requests, including "print" and "write", use the value of this index if no transaction specifiers are supplied. It is initially set to the value of the "highest transaction seen" when the user enters a meeting. The "highest transaction seen" index is a logical pointer to the highest transaction that the user has printed or written to a segment. This index is used when determining which transactions are "new".

Name: retrieve, rt

SYNTAX

```
rt trans_specs
```

FUNCTION

allows participants to retrieve specified transactions that were previously deleted with the 'delete' request. The chairman can retrieve any deleted transaction. Other participants can only retrieve transactions that they authored and subsequently deleted.

ARGUMENTS

trans_specs
: are transaction specifiers that determine which transactions are retrieved (see below). See Section 3 for complete information on transaction specifiers. Regular expressions and Person_ids cannot be used as transaction specifiers with this request.

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
: selects transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
: selects transactions entered after the date_time specified.

-before DT, -be DT
: selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
: selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
: selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

Name: retrieve_participant, rtpt

SYNTAX

```
rtpt Person_ids {-control_args}
```

FUNCTION

logically retrieves the record of participation (attendee record) of a participant in a meeting after it has been deleted with the "delete_participant" request. Only the chairman of the meeting can issue this request.

ARGUMENTS

Person_ids
are the names of the participants whose records are to be retrieved.

CONTROL ARGUMENTS

-brief, -bf
suppresses the message "Specified switch was not changed."

-long, -lg
prints the message "Specified switch was not changed." if the record for that participant has not been deleted. (Default)

set_message

Name: set_message

SYNTAX

```
set_message {-control_args}
```

FUNCTION

allows the chairman to set a message that is to be printed the first time a user enters a transaction after entering a meeting. The message is also printed the first time a user goes to the meeting after the message has changed. The message has a maximum size of 256 characters.

CONTROL ARGUMENTS

-fill, -fi
: fills the message after the user exits input mode. This is the default unless the -no_input_fill (-nif) control argument was given with the invocation of forum. The default fill width is 72, but the user can specify the fill width via the -line_length NN (-ll NN) control argument to the forum command.

-input_file pathname, -if pathname
: enters the segment identified by pathname into the proceedings. The default takes the message as it is typed in from the terminal and enters the request loop upon termination of input.

-no_fill, -nfi
: does not fill the message to the default fill width when entered. The default fills the transaction unless the -no_input_fill (-nif) control argument was specified when forum was invoked. The fill width defaults to 72, unless the -line_length NN (-ll NN) control argument was specified when forum was invoked. (Default)

-no_request_loop, nrql
: enters the message into the meeting without first entering the forum request loop. (Default except when -input_file is used.)

-request_loop, -rql
: enters the forum request loop before entering the message in the meeting.

-terminal_input, -ti
: inputs the message from the user's terminal. (Default)

TERMINAL INPUT

Unless the user makes use of the -input_file control argument, the message is built by taking lines from the user's terminal. There are three methods of exiting from terminal input mode:

. Enters the message as is, and returning to Forum request level.

\f Exits terminal input mode and invokes the qedx editor on the message.

\q Exits terminal input mode and enters the Forum request loop.

NOTES

To change a message, simply set the new message with "set_message".

A message that was not entered via the period (.) exit from terminal input mode can be entered from Forum request level via the "enter" request. See the description of the "enter" request for details.

Name: subject, sj

SYNTAX

```
sj {new_subject} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[sj {new_subject} {-control_args}]
```

FUNCTION

changes the subject of the unprocessed transaction or prints/returns the subject of the unprocessed transaction.

ARGUMENTS

new_subject
is a string formed by concatenating the non-control arguments with intervening spaces. It becomes the subject of the unprocessed transaction.

CONTROL ARGUMENTS

-default
changes the subject to that of the transaction that the unprocessed transaction is in reply to. This control argument can only be used with replies and cannot be used if new_subject is given.

-subject, -sj
begins the subject with the next argument. Useful if the first character of the subject is "-".

NOTES

When given no arguments, the subject request prints or returns the current subject of the unprocessed transaction, if any. An unprocessed transaction is a transaction that has been "built" by the user, either by exiting the "talk" request with the "\q" request or by using the "qedx" or "ted" requests; but has not yet been entered into a meeting.

Name: subsystem_name

SYNTAX

```
subsystem_name
```

SYNTAX AS AN ACTIVE REQUEST

```
[subsystem_name]
```

FUNCTION

prints the name of the subsystem; as an active request, returns the name of the subsystem. This is useful in abbrevs that are used in more than one subsystem.

Name: subsystem_version

SYNTAX

```
subsystem_version
```

SYNTAX AS AN ACTIVE REQUEST

```
[subsystem_version]
```

FUNCTION

prints the version number of the subsystem; as an active request, returns the version number of the subsystem.

switch_off

Name: switch_off, swf

SYNTAX

```
swf switch_name {-control_args}
```

FUNCTION

turns off various Forum switches.

ARGUMENTS

switch_name
is the name of the switch to reset. See "List of Switches" below.

CONTROL ARGUMENTS

-brief, bf
suppresses the "Specified switch was not changed" error message which is printed when the user tries to set a switch that is already off.

-meeting meeting_name, -mtg meeting_name
sets the switch for the meeting identified by meeting_name.

-user Person_id
sets the switch for the user identified by Person_id. Only the chairman can set another user's flag.

LIST OF SWITCHES

adjourned, adj
temporarily adjourns the meeting. When this switch is on, no users can enter the meeting. Only the chairman can reset this switch.

meeting_eligibility_messages, mtg_emsg
turns off the printing of eligibility messages for the meeting. This switch can only be reset by the chairman, and can only be reset if the site administrators allow it. (The default is site-dependent--see Appendix A.)

notify, nt
turns off the notify switch for the meeting. (This is the default.)

participating, part
turns off the participation switch for the meeting. (The default is on.)

Name: switch_on, swn

SYNTAX

```
swn switch_name {control_args}
```

FUNCTION

turns on various Forum switches.

ARGUMENTS

switch_name
is the name of the switch to set. See "List of Switches" below.

CONTROL ARGUMENTS

-brief, -bf
suppresses the "Specified switch was not changed" error message which is printed when the user tries to set a switch that is already on.

-meeting meeting_name -mtg meeting_name
sets the switch for the meeting identified by meeting_name.

-user Person_id
sets the switch for the user identified by Person_id. Only the chairman can set another user's flag.

LIST OF SWITCHES

adjourned, adj
prevents users from entering the meeting. Only the chairman can set this switch.

meeting_eligibility_messages, mtg_emsg
turns on the printing of eligibility messages for the meeting. This switch can only be set by the chairman. (The default is site dependent--see Appendix A.)

notify, nt
turns on the notify switch for the meeting. This means that an interactive message will be sent each time a transaction is entered if the user is logged in and has issued the forum_accept_notifications command (described in Section 8). (The default is off.)

participating, part
turns on the participation switch for the meeting. (This is the default.)

talk

Name: talk, t

SYNTAX

```
t {-control_args}
```

FUNCTION

allows a participant to build, and optionally enter into the proceedings of a meeting, a new transaction.

CONTROL ARGUMENTS

-brief, -bf
: suppresses the message that says that the transaction was successfully entered.

-fill, -fi
: fills the transaction after the user exits input mode. This is the default unless the -no_input_fill (-nif) control argument was given with the invocation of forum. The default fill width is 72, but the user can specify the fill width via the -line_length NN (-ll NN) control argument to the forum command.

-input_file pathname, -if pathname
: enters the segment identified by pathname into the proceedings. The default takes the transaction as it is typed in from the terminal and enters the request loop upon termination of input.

-long, -lg
: prints a message indicating that the transaction was successfully entered. (Default).

-meeting meeting_name, -mtg meeting_name
: enters the transaction into the proceedings of the meeting_name meeting. The default enters the transaction into the proceedings of the meeting the user is currently attending. The meeting_name argument can be a meeting name or a pathname.

-no_fill, -nfi
: does not fill the transaction to the default fill width when entered. The default fills the transaction unless the -no_input_fill (-nif) control argument was specified when forum was invoked. The fill width defaults to 72, unless the -line_length NN (-ll NN) control argument was specified when forum was invoked. (Default)

-no_request_loop, -nrql
: enters the transaction into the meeting without first entering the forum request loop. (Default except when -input_file is used.)

-request_loop, -rql
: enters the forum request loop before entering the transaction in the meeting.

-subject subject_string, -sj subject_string
uses subject_string as the subject, avoiding the prompt for subject. The default prompts the user for the subject. If subject_string contains spaces, it must be quoted.

-terminal_input, -ti
inputs the transaction from the user's terminal. (Default)

TERMINAL INPUT

Unless the user makes use of the -input_file control argument, the transaction is built by taking lines from the user's terminal. There are three methods of exiting from terminal input mode:

. Enters the transaction as is, returning to Forum request level.

\f Exits terminal input mode and invokes the qedx editor on the transaction.

\q Exits terminal input mode and enters the Forum request loop.

NOTES

A transaction that was not entered via the period (.) to exit from terminal input mode can be entered from Forum request level via the "enter" request. See the description of the "enter" request for details.

Name: ted

SYNTAX

```
ted {-control_arg}
```

FUNCTION

invokes the "ted" editor to edit the text of the unprocessed transaction or to build a new transaction.

CONTROL ARGUMENTS

-fill, -fi
specifies that the transaction is to be refilled upon exit from the editor. This is the default unless the -no_input_fill control argument was given to the forum command or unless the -no_fill control argument was given to the talk or reply request.

-no_fill, -nfi
specifies that the transaction is not to be refilled upon exit from the editor. (Default)

NOTES

The write (w) request must be used to reflect any changes made to the transaction back to forum. The write request can be used as often as desired to checkpoint the editing process.

If the quit (q) request is issued and the transaction has been modified since it was last written, ted will query for permission to exit. If permission is given, any changes made since the last write request will be lost. The quit-force (!q) request can be used to abort unwanted editing of the transaction without being queried.

If issued before use of the write request, the request line:

```
! 1,$dr
```

will restore the original transaction text to the buffer. If given after a write request, this request line will restore the transaction as saved by the last write request given in the buffer.

Name: unmake_public, ump

SYNTAX

ump

FUNCTION

makes all users not explicitly eligible to participate in the current meeting ineligible. Users are made eligible to participate by using the add_participant and add_project requests. Only the chairman of the meeting can use this request.

ACCESS REQUIRED

Use of this request requires access to the forum_chairman_ gate.

write

Name: write, w

SYNTAX

```
w {trans_specs} {-control_args}
```

FUNCTION

writes the selected transactions into a segment.

ARGUMENTS

trans_specs
: are transaction specifiers that determine the transactions to be written. See "List of transaction specification control arguments" below. See Section 3 for complete information on transaction specifiers. If no transaction specifiers are supplied, the current transaction is written.

CONTROL ARGUMENTS

-brief, -bf
: suppresses the message about how many transactions were written.

-by_chain
: specifies that transactions are to be grouped by transaction chain.

-extend
: writes the selected transactions to the end of the segment named either by the -pathname control argument or by default (see "Notes" below). (Default)

-fill, -fi
: specifies that the transaction is to be refilled upon exit from the editor. This is the default unless the -no_input_fill control argument was given to the forum command or unless the -no_fill control argument was given to the talk or reply request.

-format, -fmt
: specifies that transactions are to be written in a formatted fashion which includes page breaks. See "Notes on formatting" below.

-include_deleted, -idl
: selects transactions even if they have been deleted. See "Notes" below. Default is -only_non_deleted.

-initial
: writes only transactions that are at the beginning of transaction chains.

-line_length LENGTH, -ll LENGTH
: fills transactions to LENGTH characters. (Default = 80)

-long, -lg
prints the message about how many transactions were written. (Default)

-meeting meeting_name, -mtg meeting_name
selects the specified transactions from the meeting_name meeting. The default selects transactions from the current meeting.

-no_fill, -nfi
does not fill transactions when written. The default fills only transactions that were marked as unfilled when entered.

-no_format, -nfmt
specifies that transactions are to be written with no page breaks, headers, or footers. (Default)

-no_separator, -nsep
writes out transactions with a single blank line between them.

-only_deleted, -odl
selects only deleted transactions. See "Notes" below. Default is -only_non_deleted.

-only_non_deleted, -ondl
selects only non-deleted transactions. (Default)

-page_length LENGTH, -pl LENGTH
specifies the number of lines to be printed on each page. LENGTH includes the page header and footer. This control argument is only applicable if -format is given. (Default = 60)

-pathname PATH, -pn PATH
writes the transactions selected to the segment identified by PATH. The ".trans" suffix is assumed.

-reverse, -rv
writes transactions in the reverse order of that given in the transaction specifier.

-separator STR, -sep STR
appends a separator string to each transaction written to the output segment. STR is an ioa_ control string of up to 256 characters. Default is a new page ("^|").

-truncate, -tc
truncates the segment before writing the transactions to it.

LIST OF TRANSACTION SPECIFICATION CONTROL ARGUMENTS

-after DT, -af DT
selects transactions entered on or after the the date specified. The time of day is ignored.

-after_time DT, -aft DT
selects transactions entered after the date_time specified.

write

-before DT, -be DT
selects transactions entered before the date specified. The time of day is ignored.

-before_time DT, -bet DT
selects transactions entered before the date_time specified.

-between DT1 DT2, -bt DT1 DT2
selects transactions entered between the dates specified, inclusive. The times of day are ignored.

-between_time DT1 DT2, -btt DT1 DT2
selects transactions entered between the date_times specified, inclusive.

-date DT, -dt DT
selects transactions entered on the day specified.

-from Person_id, -fm Person_id
selects transactions entered by the participant named Person_id.

-subject /REGEXP/, -sj /REGEXP/
selects transactions whose subject matches the given regular expression.

-text /REGEXP/, -tx /REGEXP/
selects transactions whose text matches the given regular expression.

NOTES

If no pathname is given, the selected transactions are written to a segment in the working directory with the entryname meeting_name.trans. The user's current transaction index is set to the last transaction written.

In order to write a deleted transaction, the user must be either the chairman of the meeting or the author of the deleted transaction.

NOTES ON FORMATTING

Formatted output consists of a page header containing the current date and time, the name of the meeting, and the page number, plus a footer containing the subject of the first transaction on the page. The header and footer are separated from the text by a line of underscores and a blank line. Transactions are only split across pages if they are longer than one page. If a transaction does not fit on the remainder of the current page, a new page is always created. The -separator control argument cannot be used when formatting output.

SECTION 8

USER COMMANDS

This section contains descriptions of the Forum users' commands, presented in alphabetical order. These commands are for all Forum users, that is, for anyone who wants to find out about and attend meetings. Each description contains the name of the command (including the abbreviated form, if any), discusses the purpose of the command, and shows the correct usage. Notes and examples are included when deemed necessary for clarity.

The syntax line demonstrates the proper format to use when invoking the command. The following conventions apply in the usage line:

1. Optional arguments are enclosed in braces (e.g., {path}, {User_ids}). All other arguments are required.

2. Control arguments are identified in the usage line with a leading hyphen (e.g., {-control_args}) simply as a reminder that all control arguments must be preceded by a hyphen in the actual invocation of the command.

3. To indicate that a command accepts more than one of a specific argument, an "s" is added to the argument name (e.g., paths, {paths}, {-control_args}).

forum

Name: forum

SYNTAX AS A COMMAND

```
forum {meeting_name} {-control_args}
```

FUNCTION

The forum command enters the Forum interactive meeting system. Once the command is invoked, you are placed in the Forum subsystem, where you must use Forum requests. Forum requests are listed below under "List of Requests". (Forum requests are treated tutorially in Sections 2 through 5, and are fully described in Section 7.)

ARGUMENTS

meeting_name
: is the name or pathname of the meeting to be entered immediately upon invoking Forum. If a pathname is specified, it identifies the meeting to enter. Otherwise, Forum searches for meeting_name by using the forum search list. See "Notes on Search List" below.

CONTROL ARGUMENTS

-abbrev, -ab
: specifies that abbreviation processing should be done by the Forum request processor. If the -profile argument is not given, the user's default profile segment (>udd>Project_id>Person_id>Person_id.profile) is used.

-brief, -bf
: suppresses some messages from Forum and shortens others.

-input_fill, -ifi
: fills transactions before entering them into the proceedings of a meeting. See "Notes on Filling" below for a discussion of filling. (Default)

-line_length N, -ll N
: causes filling (fill request and default filling) to be done with the line length specified by N. (Default is 72)

-list {trans_spec}, -ls {trans_spec}
: lists the specified transactions before entering the request loop. If no transaction specifier is given, all the transactions in the meeting are listed.

-long, -lg
: uses long messages from Forum. (Default)

-meeting meeting_name, mtg meeting_name
: enters the meeting whose entryname or pathname is meeting_name. See the description of the meeting_name argument above.

-no_abbrev, -nab
specifies that abbreviation processing is not to be done by the Forum request processor. (Default)

-no_input_fill, -nif
does not fill transactions before entering them into the proceedings of a meeting. See "Notes on Filling" below for a discussion of filling.

-no_output_fill, -nof
does not fill transactions before printing them on the user's terminal. See "Notes on Filling" below for a discussion of filling. (Default)

-no_prompt
does not prompt in the request loop.

-no_start_up, -nsu
does not execute the start_up exec_com. See "Notes on start_up" below.

-output_fill, -ofi
fills transactions before printing or writing. See "Notes on Filling" below for a discussion of filling.

-profile profile_path, -pf profile_path
specifies that abbreviation processing is to be done using the profile named profile_path. If this control argument is given, the -abbrev argument need not be given.

-prompt prompt_string
sets the request loop prompt to prompt_string. (Default is "forum:")

-quit
tells Forum to process the initial request line and then return without entering the request loop (even if the initial request line is aborted).

-request string, -rq string
executes the requests in string before entering the request loop.

-start_up, -su
executes the start_up exec_com. See "Notes on start_up" below. (Default)

NOTES ON REQUESTS

Request lines use () for iteration, "" for quoting, and [] to invoke Forum active requests (listed below under "List of Active Requests").

forum

Any request line that begins with ".." is passed directly to the Multics command processor with the leading ".." stripped off. Consequently, any reference to an active function is evaluated by the Multics command processor. The "execute" (e) forum request can also make use of active strings via the square brackets ([]), but these are evaluated as active requests by Forum before the command line is passed to the Multics command processor.

NOTES ON START_UP

If the -no_start_up control argument is not given, forum will search for and execute an exec_com file of forum requests. It will look for a segment named "start_up.fmec" in the home directory, project directory, and >site in that order. The start_up will be executed before going to the initial meeting and before executing the initial request line.

NOTES ON FILLING

Transactions can be filled out to a given line length on both input and output. By default, all transactions input from the terminal are filled to a line length of 72 upon entry. The -input_fill (-if) and -no_output_fill (-nof) control arguments are the default. These control arguments specify a global attribute for the invocation of Forum, which can be overridden by control arguments associated with the various requests for inputting and outputting transactions (such as the -no_fill control argument to the "print" request). The -input_fill and -no_input_fill control arguments do not affect transactions entered using the -input_file control argument to the talk and reply requests.

LIST OF REQUESTS

Listed below are the available requests that you can use once you are in the Forum subsystem. These requests are described in detail in Section 7. Certain common requests are described tutorially in Sections 2 through 6 of this manual.

?
lists the available forum requests and active requests.

.
identifies Forum with version number; gives meeting_name if attending; gives count of new, total, last, and current transactions; and gives number of lines in the unprocessed transaction.

abbrev {-control_args}, ab {-control_args}
turns abbreviation processing on or off and changes profile segments.

add_participant NAME {-control_arg}, apt NAME {-control_args}
allows a user to participate in the meeting. (A chairman request only).

add_project NAME {-control_arg}, apj NAME {-control_args}
allows a project to participate in the meeting. (A chairman request only).

forum

answer STR {-control_args} request_line
supplies an answer to a question asked by a request. STR is the desired answer to the question and request_line is the Forum request line.

apply command_line, ap command_line
places the unprocessed transaction into a temporary segment, concatenates command_line with the pathname, and passes the result to the Multics command processor. The temporary segment is then read back in as the unprocessed transaction.

chairman {meeting_name}, cm {meeting_name}
prints the User_id (Person_id.Project_id) of the meeting's chairman.

current_meeting {-control_args}, cmtg {-control_args}
prints the name of the current meeting.

delete trans_specs, dl trans_specs
allows the chairman to delete specified transactions from the proceedings.

do {request_line} {args}, do {-control_args}
substitutes args into the request_line and passes the result to the Forum request processor. Control arguments can be -nogo to suppress execution or -long (-lg) to display expanded line before execution.

enter {-control_args}, en {-control_args}, send {-control_args}
enters the unprocessed transaction into the proceedings of a meeting.

exec_com PATH STRs, ec PATH STRs
executes an exec_com segment containing forum requests where PATH is the pathname of the exec_com segment (the ".fmec" suffix is assumed) and STRs are arguments to be passed to the program.

execute STRs, e STRs
executes STRs as a Multics command line after evaluating Forum active requests. As an active request, returns the result of evaluating strings as an Multics active string.

fill {-control_args}, fi {-control_args}
reformats transaction text to fit in a given line length.

forum_dir, fd
prints the pathname of the central forum directory.

goto meeting_name, g meeting_name
enters the user into the meeting_name meeting.

help {STR}
prints information about request names or topics. A list of available topics is produced by the list_help request.

forum

if EXPR -then LINE1 {-else LINE2}
conditionally executes one of two request lines depending on the value of an active string. EXPR is the active string that must evaluate to either "true" or "false". LINE1 is the Forum request line to execute if EXPR evaluates to "true". LINE2 is the Forum request line to execute if EXPR evaluates to "false".

list {trans_specs} {-control_args}, ls {trans_specs} {-control_args}
prints a summary of the specified transactions.

list_help {topics}, lh {topics}
prints a list of available info segments whose names include a topic string.

list_meetings {meeting_names} {-control_args} , lsm {meeting_names} {-control_args}
prints a list of selected meetings and information about them.

list_requests {-control_args}, lr {-control_args}
prints information about forum requests.

list_users {-control_args}, lsu {-control_args}
prints information about specified participants in a meeting.

make_public, mp
allows all users on the system access to the meeting. (A chairman request only.)

print {trans_specs} {-control_args}, pr {trans_specs} {-control_args}
prints selected transactions from a meeting.

qedx, qx
invokes the qedx editor on the unprocessed transaction.

quit {-control_arg}, q {-control_arg}
exits Forum.

remove_participant NAME, rpt NAME
denies the user permission to participate in the meeting. (A chairman request only.)

remove_project NAME, rpj NAME
denies users on a project permission to participate in the meeting. (A chairman request only.)

reply {trans_spec} {-control_args}, rp {trans_spec} {-control_args}
enters/builds a new transaction in a meeting that has as its subject a reference to some other transaction in the form "Re: <some other subject>", and which will be logically linked to the transaction specified by trans_spec.

reset {trans_spec} {-control_args}, rs {trans_spec} {-control_args}
resets the user's current or highest-seen transaction index to the specified transaction.

retrieve trans_specs, rt trans_specs
    retrieves specified transactions that were previously deleted with the "delete" request. Only the chairman or the author of the deleted transaction can use this request.

set_message {-control_args}
    sets a greeting message for the meeting. The default action puts the user into input mode to specify the message. (A chairman request only.)

subject {strings}, sj {strings}
    prints or modifies the subject of an unprocessed transaction. If strings are supplied, they are catenated together to become the new subject. If no strings are supplied, the current subject is printed.

subsystem_name
    prints the name of the subsystem ("forum").

subsystem_version
    prints the current version of Forum.

switch_off switch {-control_args}, swf switch {-control_args}
    turns off various Forum switches. Switches are:
        adjourned, adj
        meeting_eligibility_messages, mtg_emsg
        notify, nt
        participating, part

switch_on switch {-control_args}, swn switch {-control_args}
    turns on various Forum switches. Switches are:
        adjourned, adj
        meeting_eligibility_messages, mtg_emsg
        notify, nt
        participating, part

talk {-control_args}
    enters/builds a new transaction in a meeting.

ted
    invokes the ted editor on the unprocessed transaction.

unmake_public, ump
    allows only users granted explicit permission to participate in the meeting. (A chairman request only.)

write {trans_specs} {-control_args}, w {trans_specs} {-control_args}
    writes selected transactions to a segment.

LIST OF ACTIVE REQUESTS

[chairman {meeting_name}], [cm {meeting_name}]
    returns the Person_id.Project_id of meeting chairman.

[current_meeting {-control_args}], [cmtg {-control_args}]
returns the name of the current meeting.

[do {request_string} {args}]
returns expanded request string.

[exec_com PATH STRs], [ec PATH STRs]
executes an exec_com segment containing forum requests.

[execute STRs], [e STRs]
invokes Multics active function within forum request line.

[forum_dir], [fd]
returns absolute pathname of central forum directory.

[if EXPR -then STR1 {-else STR2}]
returns one of two character strings to the Forum request processor depending on the value of EXPR. EXPR is the active string that must evaluate to either "true" or "false". STR1 is returned if EXPR evaluates to "true". STR2 is returned if EXPR evaluates to "false".

[list_meetings {-control_args}], [lsm {-control_args}]
returns names of meetings that have new transactions.

[list_users {-control_args}], [lsu {-control_args}]
return names of participants matching given conditions.

[subject], [sj]
returns the subject of an unprocessed transaction.

[subsystem_name]
returns the name of the subsystem ("forum").

[subsystem_version]
returns the current version of Forum.

TRANSACTION SPECIFIERS

Transaction specifiers identify transactions. They are used as arguments to the forum requests that act on transactions. In addition, there are several requests and active requests that return information about transaction numbers. See Section 3 for complete information about transaction specifiers.

NOTES ON SEARCH LIST

The forum command uses the "forum" search path. For more information about search paths, see the descriptions of the search facility commands in the Commands manual. For details on the forum search path, see Section 2.

forum_accept_notifications

Name: forum_accept_notifications, fant

SYNTAX AS A COMMAND

```
fant
```

FUNCTION

enables reception of interactive messages notifying the user of new transactions being entered in meetings for the duration of this login session. Notifications are enabled for all meetings for which the user has set the notify flag. See the description of the switch_on request in Section 7 to find out how to set the notify flag.

Name: **forum_delete, fdl**

SYNTAX AS A COMMAND

```
fdl meeting_name
```

FUNCTION

deletes the specified meeting.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be deleted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be deleted. Otherwise, fdl searches for the meeting_name by using the forum search path.

ACCESS REQUIRED

The user must have "m" access to the directory containing the meeting to delete the meeting.

forum_dir

Name: **forum_dir, fd**

SYNTAX AS A COMMAND

```
fd
```

SYNTAX AS AN ACTIVE FUNCTION

```
[fd]
```

FUNCTION

prints/returns the absolute pathname of the central meeting directory.

NOTES

The central forum directory is the location for site-approved public meetings. This directory is included in the default forum search path.

forum_list_meetings

Name: forum_list_meetings, flsm

SYNTAX AS A COMMAND

```
flsm {meeting_names} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[flsm {-control_args}]
```

FUNCTION

prints a list of selected meetings on the user's terminal. For each meeting selected, information about the names of the meeting and per-user meeting attributes are listed.

ARGUMENTS

meeting_names
: are optional meeting_names. If any are supplied, information about only the specified meeting_names is printed. The star convention is supported for meeting_names. If no meeting_names are given, information for all meetings found in the forum search path is printed.

CONTROL ARGUMENTS

-absolute_pathname, -absp
: prints the absolute pathname of a meeting. The default lists the long and short meeting names only.

-adjourned, -adj
: selects information about meetings that have been adjourned.

-all, -a
: prints information about all meetings. The default prints information about the meetings in which the user is eligible to participate only.

-brief, -bf
: suppresses the message "No meetings have changed" which is printed by default if the -changes control argument is used and no meetings have changed.

-chairman {Person_id}, -cm {Person_id}
: prints information about meetings of which Person_id is chairman. If Person_id is not given, the user's Person_id is used.

-changed, -chg
: prints information about meetings in which the user is a participant and new transactions have been entered.

forum_list_meetings

-count, -ct
prints out the number of new transactions for a meeting in which the user is participating. This control argument cannot be used if forum_list is invoked as an active function.

-eligible, -elig
prints information about all meetings in which the user is eligible to participate. (Default)

-exclude meeting_names, -ex meeting_names
excludes the meetings identified by meeting_names from the output list. This control argument is designed to be used when a starname has been supplied as meeting_names in order to exclude the specified meetings from being selected. The default (-include) is to select all meetings that match the specified starname.

-from DT, -fm DT
selects meetings which have changed since the specified time. DT is any string acceptable to the convert_date_to_binary_ subroutine. The default is the current time.

-header, -he
prints the meeting header. This is the default, unless the -changes control argument was supplied.

-include meeting_names, -incl meeting_names
includes the meetings identified by meeting_names in the output list. This control argument is intended to be used when a starname has been specified for selecting meetings. (Default)

-inhibit_error, -ihe
does not print warning messages for such as things as bad meeting format and errors encountered while searching for meetings. (Default)

-long, -lg
prints the message "No meetings have changed" if the -changes control argument is used and no meetings in which the user is a participant have changed. (Default)

-no_changes, -nchg
does not list changed meetings.

-no_adjourned, -nadj
does not list adjourned meetings.

-no_header, -nhe
suppresses printing of the meeting header.

-no_inhibit_error, -nihe
prints all warning messages.

-no_notify, -nnt
    does not list meetings in which the user has the notify flag set.

-no_participating, -npart
    does not list meetings in which the user is a participant.

-no_read_only, -nro
    does not list meetings to which the user has only read access.

-notify, -nt
    lists only meetings in which the user has the notify flag set.

-participating, -part
    lists only meetings in which the user is participating. The default prints information about all meetings that the user is eligible to participate in.

-read_only, -ro
    lists just the meetings to which the user has only read access.

-user User_id
    determines participation, eligibility, and changes attributes for the user specified by User_id. User_id is in the form Person_id.Project_id.tag, where any of the components can be the character "*". If a component is omitted, it is assumed to be "*".

-verbose, -vb
    prints the chairman's User_id and current and last transactions for each meeting. This control argument cannot be used with the -changes control argument or when invoking flsm as an active function.

NOTES ON FLAGS

The output from this command may include flags that have the following interpretation:

c change flag
   new transactions have been entered in the proceedings of the meeting since the user last attended this meeting.

e eligible flag
   the user can participate in the meeting.

j adjourned flag
   the chairman has temporarily adjourned the meeting.

n notify flag
   the user has turned on the notify flag in the meeting (i.e., the user has requested online notification when new transactions are entered in the meeting).

o observer flag
   the user has only read access to the meeting.

forum_list_meetings

p participant flag
the user is a participant in meeting (i.e., has gone to the meeting at least once).

r removed flag
the user has removed himself from participation in the meeting.

The flags corresponding to the selection criteria are not shown (i.e., if the user is selecting meetings to which he is eligible, the "e" flag is not printed, it is assumed).

forum_list_users

Name: forum_list_users, flsu

SYNTAX AS A COMMAND

```
flsu {meeting_name} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[flsu {meeting_name} {-control_args}]
```

FUNCTION

prints a list of selected participants of a meeting. For each participant selected, the Person_id, Project_id, current transaction number (last one seen), the date/time last attended, and several flags are listed. As an active function it returns a list of Person_ids.

ARGUMENTS

meeting_name
: is the name or pathname of the meeting for which participants are to be listed. Either this argument or the -meeting control argument (but not both) must be given.

CONTROL ARGUMENTS

-all, -a
: lists all participants, including those who have been removed from the meeting.

-attending, -at
: lists only participants who are currently attending this meeting.

-brief, -bf
: suppresses the message "No participants were selected."

-eligible, -elig
: prints a list of users and projects that are eligible to attend the meeting. Read-only eligibility is denoted by an asterisk (*).

-header, -he
: prints the header. (Default)

-long, -lg
: prints the "No participants were selected." message if this is the case. (Default)

-meeting meeting_name, -mtg meeting_name
: lists participants of the meeting_name meeting. The meeting_name argument can be either a name or a pathname of a meeting. The default lists participants of the current meeting.

-no_header, -nhe
suppresses printing of the name_list header.

-no_read_only, -nro
does not list read-only participants.

-notify, -nt
lists only participants with the notify flag on.

-project Project_ids, -pj Project_ids
lists only information about participants on the specified project. All arguments following -project until the next control argument are taken as Project_ids.

-read_only, -ro
lists just read-only participants.

-seen transaction_number
lists only participants who have read the specified transaction.

-sort TYPE
sorts the output by TYPE. TYPE is either "name" for sorting by the Person_id of participants, or "date_time_attended" (dta) for sorting by the time the participant last attended the meeting. This control argument can not be used for the active function.

-totals, -tt
prints only the total number of participants selected.

-unseen transaction_number
lists only participants who have not read the specified transaction.

-user Person_ids
prints only information about the named participants. All arguments following -user until the next control argument are taken as Person_ids.

NOTES

The displayed flags have the following meanings:

r the participant has been removed from the meeting and is therefore no longer a participant.

n the user has the notify flag turned on.

o the user is an observer and cannot enter transactions.

forum_refuse_notifications

Name: forum__refuse__notifications, frnt

SYNTAX AS A COMMAND

```
frnt
```

FUNCTION

disables reception of interactive messages notifying the user of new transactions being entered in meetings.

SECTION 9

DUTIES OF THE CHAIRMAN

This section shows how to create and maintain meetings. For detailed descriptions of the chairman's commands referred to here, see Section 10.

CREATING A MEETING

To create a meeting, use the forum_create (fcr) command. Forum asks you for all the necessary information, step by step. In the following example, assume that the user named Fox on the ProjA project wants to create a meeting in her working directory and that she wants it to be a public meeting.

```
! fcr

  Enter pathname of meeting directory (carriage return for
  working_dir):     ! <CR>

  Please enter long meeting name (<20 characters):     ! home_computers

  Now enter abbreviated meeting name:     ! hc

  Should the meeting be public?     ! yes

  Do you want to enter a chairman message (? for explanation)?  ! yes
  Message:
  ! Welcome to the home_computers meeting.
  ! .

  The home_computers meeting has been established in >udd>ProjA>Fox.
  You must now enter the first transaction in the home_computers meeting,
  which will act as as an introduction.
  The meeting is public.  Welcome to the home_computers meeting.
  Transaction:
! This meeting is designed to allow anyone interested in personal
! computing to discuss any issues related to such.  Participants are
! asked to indicate which home computer they have.  The meeting is
! open to the public.
! .
  Transaction [0001] entered in >udd>ProjA>Fox>home_computers meeting.
  r 10:30 2.096 87
```

Now, if you reenter Forum and go to the home_computers meeting, Forum tells you that you are the chairman. The subject is automatically added to the first transaction since the first transaction is always the introduction.

```
! forum hc
home_computers meeting:  0 new, 1 last (You are the chairman).
forum:  ! p 1
[0001] (4 lines) Fox.ProjA 04/28/82 1030.2 est Wed eve
Subject:  Reason for this meeting
<same text as shown in above example>
---[0001]---
```

If you decide that you want to invite private participation only, you can specify exactly who can attend. When Forum asks whether the meeting should be public, simply reply no. Forum then offers to let you name individuals and/or whole projects as eligible participants. Individuals are specified by their Person_ids, and projects are specified by their Project_ids. The example below illustrates the questions asked by Forum. This sample meeting is created by Fox and is named basic_problems (basprob).

```
  .
  .
  .
Should the meeting be public?    ! no

Should specified individuals be allowed to participate?    ! yes

Now please type Person_ids of attendees when prompted.
Signal the end of the list by typing a period only.

Person_id:    ! Montgomery
Person_id:    ! Manselle
Person_id:    ! .

Should specified projects be allowed to participate?    ! yes

Now please type Project_ids when prompted.
Signal the end of the list by typing a period only.

Project_id:    ! ProjA
Project_id:    ! ProjB
Project_id:    ! .
```

Person_ids and Project_ids are entered in a list, each followed by a carriage return (RETURN). When your list is complete, type a "." on a line by itself.

Now that you have selected the possible eligible participants, you are prompted for the chairman's message, the meeting is established, and you enter the first transaction.

```
Do you want to enter a chairman message (? for explanation)?  ! yes
Message:
! This is a developmental meeting
! open to engineers only.
! .
The basic_problems meeting has been established in >udd>ProjA>Fox.
You must now enter the first transaction in the basic_problems
meeting, which will act as an introduction
This is a developmental meeting open to engineers only.
Transaction:
!
! <transaction text>
! .
Transaction [0001] entered in >udd>ProjA>Fox>basic_problems meeting.
r 10:45 3.175 90
```

MAINTAINING A MEETING

As chairman, you can adjust the list of users who are eligible to participate in your meeting, adding and deleting users and projects as necessary at any time. The -read_only control argument to the forum_add_participant and forum_add_project commands (described below) lets you specify that certain users and/or groups of users be allowed only to read transactions, not to enter them.

You also have the right to use certain Forum requests to delete offensive or irrelevant transactions, giving proper notice to the participants, and to retrieve transactions.

As chairman, whether you created the meeting or inherited the chairmanship, you alone have the right to alter any aspects of the meeting.

Adding a Participant

It is simple to add a participant to the list of users who are eligible to attend your meeting. The forum_add_participant (fapt) command adds users to the list, one at a time. To add the user White to the list of participants for the home_computer meeting that has been added to your search rules, type:

```
! fapt home_computer White
White added to the home_computer meeting.
r 08:42 0.247 66
```

Participants can only be added one at a time.

Adding a Project

Adding a project name to your list of eligible participants is done the same way as adding a user, only with the forum_add_project (fapj) command. (You can only add one project at a time.) The example below shows two projects being added to the list of eligible participants.

```
! fapj home_computer ProjB
Project ProjB added to the home_computer meeting.
r 08:53 0.290 66
! fapj home_computer Engr
Project Engr added to the home_computer meeting.
r 08:54 0.187 6
```

Making a Meeting Public

The forum_make_public (fmp) command opens a meeting to public participation, thereby overriding any explicit eligibility requirements that may have previously been set. You can, however, remove participants and projects from a public meeting with the forum_remove_participant and forum_remove_project commands (described below).

```
! fmp home_computer
Meeting home_computer is now public.
r 08:57 0.319 1
```

Making a Meeting Private

Suppose you want to make a previously public meeting private and deny participation rights to an individual or a project.

The forum_unmake_public (fump) command makes a formerly public meeting into a private one.

```
! fump home_computer
Meeting home_computer is no longer public.
r 08:06 0.215 63
```

Now that the meeting is private, you must explicitly make users eligible to attend, or no one will be able to. Use the forum_add_participant and forum_add_project commands as shown above.

Removing a Participant

Three levels of removal (access denial, logical removal, and physical removal) allow you to barr participants in degrees of permanency.

On the top level, you can deny access to a meeting for a participant with the forum_remove_participant (frpt) command. This command removes the user's record of participation or attendee record (showing User_id, last time attended, and last transaction seen) from the list of participants that the "list_users" request normally displays. Any user can, however, see the names of removed participants by using the "list_users -all" request, which includes in its display removed participants (indicated by the "r" flag).

To remove a participant from the list of eligible participants, use the forum_remove_participant (frpt) command:

```
! frpt home_computer White
White has been removed from the home_computer meeting.
r 08:10 0.347 86
```

You can go one step further and logically remove (i.e., conceal from public view) that user's record of participation in your meeting with the "delete_participant" request.

```
forum: ! dlpt White

forum:
```

When you issue this request, Forum does not print an acknowledgment--the participant is simply removed. Now the user's record of participation is invisible to everyone but the chairman. You can still change your mind and retrieve the record with "retrieve_participant", which replaces that user's record in the list of users.

To physically (and finally) remove a participant's record, use the "expunge" request which physically removes all deleted participants from a meeting. Note that participants to be expunged must first be deleted with the delete_participant request.

```
forum:  ! expunge -participants
forum (expunge): Expunging the meeting will destroy all records
of deleted participants.  Do you really want to expunge the
home_computer meeting?  ! yes
4 participants expunged.

forum:
```

This request allows you to reclaim space and save your quota. This is necessary if your meeting is too big and in danger of running out of storage space (approaching the limit of 500 attendees.) The "expunge" request also validates the internal structure of the meeting and deletes damaged items, but it is time-consuming and generally only done to save a meeting.

Removing a Project

To remove a Project_id from the eligible list, use the forum_remove_project (frpj) command:

```
! frpj home_computer ProjB
Project ProjB removed from home_computer meeting.
r 08:14 0.421 86
```

Deleting Transactions

Occasionally you will need to delete transactions. As chairman, you have the power to remove offensive or irrelevant transactions. Two levels of deletion (logical removal and physical removal) allow you to remove transactions in degrees of permanency. First, the Forum "delete" (dl) request logically removes specified transactions from a meeting. You can delete a range of transactions, if necessary. In the example below, the chairman deletes the transaction while attending the meeting:

```
forum:  ! dl 7

forum:
```

After Forum deletes the transaction, it puts you back into the request loop. Thereafter, when any attendee attempts to specify the deleted transaction, Forum answers that the transaction has been deleted.

```
forum:  ! ls 7
forum (list): The specified transaction has been deleted.  Transaction 7.
forum (list): No transactions were selected.

forum:
```

The first Forum message in the above example explains that transaction 7 has been deleted and will therefore not be found by the "list" request. Since no transactions other than transaction 7 were specified in the request line, the "list" request has not selected any transactions and therefore printed the second message. If the request had specified two transactions (e.g., "ls 7:8"), Forum would have listed only transaction 8.

Remember that position 7 in the proceedings is permanently filled, even though the transaction has been deleted. The next transaction entered is numbered 8, and transaction 7, although it no longer appears in the proceedings, is saved elsewhere and can be retrieved at any time with the "retrieve" request (see below).

```
forum:  ! ls 1:8
Trans#  Lines   Date     Time   Author         Subject
[0001]   (5)   05/18/81  03:20  Brooks.ProjA   Reason for this meeting
[0002]   (3)   05/20/81  03:46  Davis.YAK      Fortran for a Begi<More>
[0003]   (3)   05/21/81  06:49  Smith.ProjB    Other Meetings
[0004]           .
[0005]           .
[0006]           .
[0008]           .

forum:
```

Notice that there is no transaction [0007] listed. As chairman, you usually enter another transaction yourself to explain the disappearance of other transactions, explaining why the deletion was necessary.

Second, to physically (and finally) remove deleted transactions from the meeting, use the "expunge" request:

```
forum: ! expunge -transactions
forum (expunge: Expunging the meeting will destroy all records
of deleted transactions.  Do you really want to expunge the
home_computer meeting?  ! yes
6 transactions expunged.

forum:
```

Note again that only deleted transactions can be expunged.

Retrieving Participants and Transactions

The "retrieve_participant" (rtpt) request retrieves a user's record of participation after you have deleted that record with the "delete_participant" request. The user's record of participation is restored to the list of the meeting's participants.

```
forum: ! rtpt Black

forum:
```

The "retrieve" (rt) request retrieves transactions that you have previously removed from the proceedings. You can retrieve a range of transactions, if necessary, but regular expressions and Person_ids can not be used as transaction specifiers with this request. To print a list of the deleted transactions in the current meeting, type "list -only_deleted". To retrieve transaction [0007], type:

```
forum:  ! rt 7

forum:
```

Now when you list the proceedings of the meeting, transaction [0007] appears in its original slot in the list and has been restored in its entirety to the meeting.

Setting a Message

The "set_message" request allows you to enter a message that is printed each time a user enters the meeting or enters a transaction from elsewhere within Forum. You can also use this request to change the message that you have previously set. Some messages set by chairmen are greetings ("Welcome to the Television meeting") while other messages describe restrictions or warnings ("This is a developmental meeting open to Honeywell employees only"). The message has a maximum allowable size of 256 characters.

Enter the meeting that you want to set a message in. If you have already entered the message into a segment, use the "-input_file pathname" control argument with "set_message" to set the message that is contained in that segment. Otherwise, type only "set_message", and Forum prompts you for the message, which you must end in one of the ways described for exiting from terminal input mode.

If you have written your message into a segment named "greeting" you would set the message this way (assume that you are already attending the Television meeting):

```
forum:  ! set_message -if greeting
Use the "enter" request to enter the message.

forum:  ! enter

forum:
```

The message has been set.

Suppose you want to build the message from within Forum:

```
forum:  ! set_message
Message:
! Welcome to the Television meeting.
! This meeting is for discussing and rating
! television shows shown at prime-time.
! Please try to be as brief as possible.
! .

forum:
```

Again, the message has been set.

Printing Eligibility Messages

The eligibility message for a meeting, as shown in Section 2, greets the user upon entrance to the meeting. Site administrators decide whether or not these messages are globally printed for all meetings on the system. If site administration turns them on globally, you (the chairman) can then either go along with the system-wide printing of these messages or turn printing off for the meeting that you chair.

The "switch_on" (swn) and "switch_off" (swf) requests accept the switchname "meeting_eligibility_messages" (mtg_emsg). To turn off printing of the message (only if your site administration has turned printing on), type:

```
forum:  ! swf mtg_emsg
```

To turn printing back on, type:

```
forum:  ! swn mtg_emsg
```

NAMING A NEW CHAIRMAN

To pass on chairmanship of the meeting, use the "chairman" (cm) request. The new chairman must be specified by his/her Person_id.Project_id.

```
forum:  ! cm -set JohnB.ProjA

forum (chairman):  Do you really want to change the chairman of the
Television meeting to JohnB.ProjA?  ! yes

forum:
```

DELETING A MEETING

If the reason for your meeting was a question that has been resolved, or was to discuss and prepare for an event that is now over, or in any event is no longer valid, you will want to delete your meeting. Of course you will want to give reasonable notification to the participants of the meeting, perhaps through a transaction or by setting a message. The forum_delete (fdl) command deletes a meeting:

```
! fdl home_computers
r 08:23 0.307 2
```

SECTION 10

CHAIRMAN COMMANDS

The commands in this section are intended to be used only by the chairman, or maintainer, of the meeting. From the time that the meeting is created, its creator, the chairman, is the only user who can make changes regarding the eligibility of participants. (For a step-by-step breakdown of the duties of a chairman, see Section 9.)

The descriptions of the Forum administrative commands are presented here in alphabetical order. Each description contains the name of the command (including the abbreviated form, if any), discusses the purpose of the command, and shows the correct usage. Notes and examples are included when deemed necessary for clarity.

forum_add_participant

Name: forum_add_participant, fapt

SYNTAX AS A COMMAND

```
fapt meeting_name Person_id {-control_arg}
```

FUNCTION

makes the person identified by Person_id eligible to attend the meeting_name meeting.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be adjusted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be adjusted. Otherwise, fapt searches for the meeting_name by using the forum search list.

Person_id
: is the Person_id of the participant to be added.

CONTROL ARGUMENTS

-read_only, -ro
: allows the added participant to read transactions but not enter them.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

Name: forum_add_project, fapj

SYNTAX AS A COMMAND

```
fapj meeting_name Project_id {-control_arg}
```

FUNCTION

makes users from the project identified by Project_id eligible to attend the meeting_name meeting.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be adjusted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be adjusted. Otherwise, fapj searches for the meeting_name by using the forum search list.

Project_id
: is the Project_id for the project to be admitted to the meeting.

CONTROL ARGUMENTS

-read_only, -ro
: allows users on the added project to read transactions but not enter them.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

Name: forum_create, fcr

SYNTAX AS A COMMAND

```
fcr
```

FUNCTION

creates a new meeting with the user as chairman. It engages the user in a dialogue, prompting for all information necessary for the creation of a new meeting including long and short meeting names, participant information, and the first transaction in the meeting.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

See Section 9 for a detailed discussion including a complete example showing the creation of a meeting with this command.

forum_make_public

Name: forum_make_public, fmp

SYNTAX AS A COMMAND

```
fmp meeting_name {control_arg}
```

FUNCTION

opens the meeting_name meeting to public participation.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be made public. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be made public. Otherwise, fmp searches for the meeting_name by using the forum search list.

CONTROL ARGUMENTS

-read_only, -ro
: allows the public to read transactions but not enter them.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

forum_remove_participant

Name: forum_remove_participant, frpt

SYNTAX AS A COMMAND

```
frpt meeting_name Person_id
```

FUNCTION

makes the person identified by Person_id ineligible to attend the meeting_name meeting.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be adjusted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be adjusted. Otherwise, frpt searches for the meeting_name by using the forum search list.

Person_id
: is the Person_id of the user to be removed from participation in the meeting.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

forum_remove_project

Name: forum_remove_project, frpj

SYNTAX AS A COMMAND

```
frpj meeting_name Project_id
```

FUNCTION

denies eligibility to all participants who are currently eligible to attend by virtue of their project participation alone (not by virtue of their Person_id) from the meeting_name meeting.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be adjusted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be adjusted. Otherwise, frpj searches for the meeting_name by using the forum search list.

Project_id
: is the Project_id of the project whose members are to be denied participation rights if they lack explicit Person_id participation rights.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

forum_unmake_public

Name: forum__unmake__public, fump

SYNTAX AS A COMMAND

```
fump meeting_name
```

FUNCTION

removes all participants from the meeting_name meeting who are participating by virtue of the public attribute. All users who are not participating because of Project_id or Person_id rights are removed.

ARGUMENTS

meeting_name
: is the pathname or entryname of the meeting to be adjusted. Pathnames contain greater-than (>) or less-than (<) characters. If a pathname is specified, it identifies the meeting to be adjusted. Otherwise, fump searches for the meeting_name by using the forum search list.

ACCESS REQUIRED

The user must have "e" access to the forum_chairman_ gate to use this command.

NOTES

The user invoking this command must be the chairman of the meeting_name meeting.

APPENDIX A

ADMINISTRATIVE INFORMATION

System administrators can change the attributes of any meeting by the use of the forum_admin command. The forum_admin command provides the necessary administrative backup in the event a chairman is no longer able to function in that capacity, as well as the ability to determine system-wide policies regarding optional Forum features.

In order to use Forum at a site, you must do a few things other than install the software and associated info segments. The central meeting directory and notifications database must be created, and there are several "customizing" features regarding access control that a site may want to change.

INSTALLATION INSTRUCTIONS

The Central Forum Directory

This directory is the repository for public site-approved meetings. It is known to the forum software and is in the default "forum" search path. The pathname of the directory is >site>forum_dir. This directory should be created and given an ACL term of "s *.*.*" along with the "sma" ACL terms given to administrators entrusted with creating meetings there.

If for some reason, a site wishes to use some other directory as the central directory, the following steps should be taken:

1. Change the assignment of forum_external_data.central_directory in the segment forum_data_.cds to the desired path and recompile it.

2. Initiate the new copy of forum_data_ and recompile the segment forum_search_list_default_.cds in bound_forum_.s.archive

3. Update forum_search_list_default_ in bound_forum_.archive and rebind bound_forum_.

4. Install the new source/object/executable versions of bound_forum_ and forum_data_.

The Notifications Database

The Forum subsystem maintains a database of users who are currently accepting notifications of new transactions (see Section 4 for an explanation of notifications). This segment resides in the central forum directory and must be created before notifications can be used. An administrator who has access to the forum_admin_ gate should create this segment with the command:

```
! forum_admin init_notifications
```

(See the "Gate Access" below for information on the forum_admin_ gate.)

Gate Access

The Forum subsystem has three gates: forum_, forum_chairman_, and forum_admin_. The forum_ gate is essential to use the subsytem and should have its access set to "re *.*.*". The forum_chairman_ gate controls who may create, set access for, and delete meetings. It is recommended that the access for this gate be set to "re *.*.*", but if the site desires to restrict a portion of the user community from performing these functions, those users should be given null access to this gate.

The forum_admin_ gate is used to perform administrative functions for Forum and to perform chairman functions on meetings (such as changing access and changing the chairman) by users other than the chairman of those meetings. This is a highly-privileged gate. The default access to this gate is "re *.SysMaint.* null *.*.*".

Eligibility Messages

In order to keep participants informed of the audience of a particular meeting, Forum provides an optional facility that tells participants how many projects and users may read a meeting, and warns the participant each time the access to the meeting changes. The chairman of the meeting normally invokes this facility, but it can be manipulated by users with access to the forum_admin_ gate as follows. The command:

```
! forum_admin switch_on print_eligibility_messages
```

will cause Forum to print eligibility messages for every meeting on the system unless specifically exempted by the following two switches.

The command:

```
! forum_admin switch_on chairman_set_eligibility_msgs
```

allows the chairman to turn eligibility messages off even if the print_eligibility_messages switch is on, and the command:

```
! forum_admin switch_(on off) meeting_eligibility_messages PATH
```

will turn eligibility messages for the meeting identified by PATH on or off regardless of the setting of the system-wide switches.

AIM

Forum is a single-class AIM facility. Meetings are single-class segments. All participants must be logged in at the same authorization, which must be the access class of the meeting. Different meetings, however, may be at different access classes. The access class of a meeting is the authorization of the chairman at the time the meeting was created.

Name: forum_admin

SYNTAX AS A COMMAND

```
forum_admin key {arguments}
```

FUNCTION

allows system administrators to change attributes of meetings that they do not chair. This command is also used to perform certain administrative functions such as initializing the forum notifications database.

ARGUMENTS

key
is any key described below in "List of Keys."

arguments
are the arguments for individual keys described below.

LIST OF KEYS

add_participant path Person_id
makes the meeting whose pathname is "path" accessible to Person_id. Person_id can be one or more characters and cannot contain a ".".

add_project path Project_id
makes the meeting whose pathname is "path" accessible to users on the project named by Project_id. Project_id can be one or more characters and cannot contain a ".".

change_chairman path new_chairman
displays the User_id of the current chairman of the meeting whose pathname is "path" and then shifts the chairmanship to new_chairman. new_chairman must be of the form Person_id.Project_id.

delete path
deletes the meeting whose pathname is "path".

init_notifications
creates an empty notifications database segment in the central forum directory.

remove_participant path Person_id
makes the meeting whose pathname is "path" not accessible to Person_id. Person_id can be one or more characters and cannot contain a ".".

remove_project path Project_id
makes the meeting whose pathname is "path" not accessible to users on the project named by Project_id. Project_id can be one or more characters and cannot contain a ".".

switch_off switchname, swf switchname
turns the specified switch off. See "List of Switches" below.

switch_on switchname, swn switchname
turns the specified switch on. See "List of Switches" below.

LIST OF SWITCHES

adjourned path, adj path
temporarily adjourns the meeting specified by path. When this switch is on, no users may enter the meeting.

chairman_set_eligibility_msg, cm_set_emsg
allows the chairman to override the print_eligibility_messages switch.

meeting_eligibility_messages path, mtg_emsg path
turns on eligibility messages for the meeting named by "path".

print_eligibility_messages, pemsg
turns on eligibility messages for all meetings on the system.

ACCESS REQUIRED

This command requires access to the forum_admin_ gate. In addition, the delete and access-changing keys require at least "m" access to the directory containing the meeting, as does setting a switch on a meeting if access must be forced. Setting global switches requires "rw" access to forum_data_, and the init_notifications key requires "sma" access to the central forum directory.

APPENDIX B

SUBROUTINE DESCRIPTIONS

Name: forum_

The forum_ subroutine provides a subroutine interface to the Forum interactive meeting system.

Entry: forum_$accept_notifications

This entry turns on reception of forum notifications by the calling process.

USAGE

```
declare forum_$accept_notifications entry (fixed bin(35));

call forum_$accept_notifications (status);
```

ARGUMENTS

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$no_notify_seg
The notifications database could not be initiated.

Entry: forum_$check_user

This entry is used to determine if a user is a participant in a Forum meeting.

USAGE

```
declare forum_$check_user entry (fixed bin, char(*), fixed bin,
      fixed bin(35));

call forum_$check_user (forum_idx, user_name, last_entered, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

user_name
The Person_id of the user to be checked. (Input)

last_entered
is the index of the most recent transaction entered by this user, or 0 if this user has never entered a transaction. (Output)

status
is a standard status code. (Output)

Entry: forum_$close_forum

This entry closes a meeting opened by forum_$open_forum.

USAGE

```
declare forum_$close_forum entry (fixed bin, fixed bin(35))

call forum_$close_forum (forum_idx, status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. This argument will be set to zero if the meeting is successfully closed. (Input/Output)

status
: is a standard status code. It can have the following value:

  forum_error_table_$invalid_forum_idx
  : forum_idx does not identify an open meeting.

Entry: forum_$convert_attendee_idx

This entry is used to convert an attendee_idx into a user name.

USAGE

```
declare forum_$convert_attendee_idx entry (fixed bin, fixed bin,
      char(*), fixed bin(35));

call forum_$convert_attendee_idx (forum_idx, attendee_idx, user_name,
      status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

attendee_idx
: is the index of a participant in a meeting, as placed in Forum interprocess messages. (Input)

user_name
: is the name of the user with the specified attendee_idx, in the form Person_id.Project_id. (Output)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$invalid_attendee_idx
The supplied attendee index does not correspond to a valid participant record in this meeting.

Entry: forum_$delete_forum

Deletes a Forum meeting.

USAGE

```
declare forum_$delete_forum entry (char(*), char(*), fixed bin(35));

call forum_$delete_forum (dirname, entryname, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

status
is a standard status code. (Output)

ACCESS REQUIRED

The user must have modify permission on the containing directory.

Entry: forum_$enter_trans

This entry is used to enter a transaction into a Forum meeting.

USAGE

```
declare forum_$enter_trans entry (fixed bin, char(*), fixed bin,
      char(*), bit(1) aligned, fixed bin, fixed bin(35));

call forum_$enter_trans (forum_idx, text, previous_trans, subject,
      filled_sw, trans_idx, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

text
is the text of the transaction to be entered. (Input)

previous_trans
is the index of the transaction that this transaction is in reply to, or 0 if it is the start of a new chain. (Input)

subject
is the subject of the transaction to be entered. (Input)

filled_sw
is set if the transaction should not be filled on output. (Input)

trans_idx
is the index of the transaction entered. (Output)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$read_only
The caller does not have access to enter transactions in the meeting.

Entry: forum_$forum_info

This entry is used to obtain information about a Forum meeting.

USAGE

```
declare forum_$forum_info entry (char(*), char(*), char(*),
      fixed bin(71), ptr, fixed bin(35));

call forum_$forum_info (dirname, entryname, access_name, access_time,
      forum_info_ptr, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

access_name
is the Person_id.Project_id.tag for which access is to be calculated. If blank, the access_name of the caller is used. The per-user information returned is for the Person_id portion of access_name. (Input)

access_time
changed information is calculated from the specified date-time. Usually, "clock ()" should be passed. (Input)

forum_info_ptr
is a pointer to a copy of the forum_info structure described below. (Input)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$not_eligible
The user does not have access to obtain information about transaction counts and changes, but the chairman name is still valid in this case.

NOTES

The forum_info structure is declared in the include file forum_info.incl.pl1 and has the following structure:

```
dcl 1 forum_info              aligned based (forum_info_ptr),
      2 version               fixed bin,
      2 forum_uid             bit(36),
      2 chairman              unaligned,
        3 username            char(20),
        3 project             char(9),
        3 pad                 char(3),
      2 attendee_count        fixed bin,
      2 removal_count         fixed bin,
      2 transaction_count     fixed bin,
      2 deletion_count        fixed bin,
      2 last_seen_trans_idx   fixed bin,
      2 last_time_changed     fixed bin(71),
      2 last_time_attended    fixed bin(71),
      2 changes_count         fixed bin,
      2 flags                 unaligned,
        3 eligible            bit(1),
        3 mbz1                bit(1),
        3 removed             bit(1),
        3 notify              bit(1),
        3 attending           bit(1),
        3 mbz2                bit(2),
        3 read_only           bit(1),
        3 adjourned           bit(1),
        3 mbz3                bit(27);
```

STRUCTURE ELEMENTS

version
is the version number of this structure. This must be set to the constant forum_info_version_1, which is declared in the include file.

forum_uid
is the unique identifier of the meeting.

username
is the Person_id of the chairman of the meeting.

project
is the Project_id of the chairman of the meeting.

pad
is unused.

attendee_count
is the total number of participants in the meeting.

removal_count
is the number of participant records marked as deleted.

transaction_count
is the total number of transactions entered before access_time.

deletion_count
is the number of those transactions that have been deleted.

last_seen_trans_idx
is the number of the most recently entered transaction that has been read by this user.

last_time_changed
is the time the most recent transaction was entered.

last_time_attended
is the time that the user last attended the meeting.

changes_count
is the number of transactions entered since access_time that the user has not seen.

eligible
is set if the user is eligible to attend the meeting.

mbz1
is unused.

removed
is set if the user has turned off the participating switch.

notify
is set if the user has turned on the notify (.

attending
is set if the user is currently attending the meeting.

mbz2
is unused.

read_only
is set if the user may not enter transactions in the meeting.

adjourned
is set if the chairman has adjourned the meeting.

mbz3
is unused.

Entry: forum_$forum_info_idx

This entry is used to obtain information about a Forum meeting. It is the same as the forum_info entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_$forum_info_idx entry (fixed bin, char(*), fixed bin(71),
      ptr, fixed bin(35));

call forum_$forum_info_idx (forum_idx, access_name, access_time,
      forum_info_ptr, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

access_name
is the Person_id.Project_id.tag for which access is to be calculated. If blank, the access_name of the caller is used. The per-user information returned is for the Person_id portion of access_name. (Input)

access_time
changed information is calculated from the specified date-time. Usually, "clock ()" should be passed. (Input)

forum_info_ptr
is a pointer to a copy of the forum_info structure described under the forum_$forum_info entry point. (Input)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$not_eligible
The user does not have access to obtain information about transaction counts and changes, but the chairman name is still valid in this case.

Entry: forum_$forum_limits

This entry is used to obtain various pieces of information about an open Forum meeting. This entry point is obsolete. The forum_$real_forum_limits entry point should be used instead.

USAGE

```
declare forum_$forum_limits entry (fixed bin, fixed bin, fixed bin,
      fixed bin, fixed bin, bit(36) aligned, fixed bin(35));

call forum_$forum_limits (forum_idx, last_seen, first_trans, last_trans,
      new_trans_count, flags, status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

last_seen
: is the index of the most-recently entered transaction that the user has seen, as set by forum_$set_last_seen_idx. (Output)

first_trans
: is the index of the oldest transaction in the meeting. (Output)

last_trans
: is the index of the most recent transaction in the meeting. (Output)

new_trans_count
: is the number of transactions entered since last_seen by users other than the caller. (Output)

flags
: is a collection of switches as described below. (Output)

status
: is a standard status code. (Output)

NOTES

The flags argument is declared in forum_flags.incl.pl1 and has the following strucure:

```
dcl 1 forum_flags              aligned based (addr (forum_flags_word)),
      2 chairman               bit(1) unaligned,
      2 read_only              bit(1) unaligned,
      2 print_cm_message       bit(1) unaligned,
      2 print_acl_message      bit(1) unaligned,
      2 acl_has_changed        bit(1) unaligned,
      2 adjourned              bit(1) unaligned,
      2 mbz                    bit(30) unaligned;
```

STRUCTURE ELEMENTS

chairman
is set if the user is the chairman of the meeting.

read_only
is set if the user may not enter transactions in the meeting.

print_cm_message
is set if the chairman message has changed since the last time the user saw it.

print_acl_message
is set if the access eligibility message should be printed.

acl_has_changed
is set if the access to the meeting has changed since the last time the user was informed that it had changed.

adjourned
is set if the chairman has adjourned the meeting.

mbz
is not used.

Entry: forum_$get_forum_path

This entry is used to determine the actual name of the meeting instead of the name of the link used to access it.

USAGE

```
declare forum_$get_forum_path entry (char(*), char(*), char(*), char(*),
      fixed bin(35));

call forum_$get_forum_path (dirname, entryname, forum_dirname,
      forum_entryname, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

forum_dirname
is the pathname of the directory where the meeting actually resides. (Output)

forum_entryname
is the primary name of the meeting. (Output)

status
is a standard status code. (Output)

Entry: forum_$get_forum_path_idx

This entry is used to determine the actual name of the meeting instead of the name of the link used to access it. It is the same as the get_forum_path entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_$get_forum_path_idx entry (fixed bin, char(*), char(*),
      fixed bin(35));

call forum_$get_forum_path_idx (forum_idx, forum_dirname,
      forum_entryname, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

forum_dirname
is the pathname of the directory where the meeting actually resides. (Output)

forum_entryname
is the primary name of the meeting. (Output)

status
is a standard status code. (Output)

Entry: forum_$get_message

This entry is used to read out the greeting message set by the chairman using the forum_$set_message entry.

USAGE

```
declare forum_$get_message entry (fixed bin, char(*), fixed bin(35));

call forum_$get_message (forum_idx, message, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

message
is the text of the message, this can contain at most 256 characters. (Output)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$no_message
There is no chairman greeting message for this meeting.

Entry: forum_$list_forum_acl

This entry is used to list the ACL of a Forum meeting.

USAGE

```
declare forum_$list_forum_acl entry (char(*), char(*), ptr, ptr,
      fixed bin, fixed bin(35));

call forum_$list_forum_acl (dirname, pathname, area_ptr, acl_ptr,
      acl_count, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

area_ptr
points to an area where the segment_acl structure is to be allocated. (Input) It can not be null.

acl_ptr
points to the allocated segment_acl structure. This structure is described in the hcs_$add_acl_entries subroutine. (Output)

acl_count
is the number of ACL entries in the segment_acl structure. (Output)

status
is a standard status code. (Output)

Entry: forum_$list_users

This entry is used to extract information about the participants in a Forum meeting.

USAGE

```
declare forum_$list_users entry (char(*), char(*), ptr, ptr,
      fixed bin(35));

call forum_$list_users (dirname, entryname, area_ptr, user_list_ptr,
      status);
```

ARGUMENTS

dirname
: is the pathname of the directory containing the meeting. (Input)

entryname
: is the name of the meeting. The ".control" suffix is required. (Input)

area_ptr
: is a pointer to an area where the user_list structure is to be allocated. (Input) It can not be null.

user_list_ptr
: is a pointer to the user_list structure described below. (Output)

status
: is a standard status code. (Output)

NOTES

The user_list structure is declared in the include file forum_user_list.incl.pl1 and has the following structure:

```
dcl 1 user_list                based (user_list_ptr) aligned,
      2 version                fixed bin,
      2 chairman               aligned,
        3 person_id            char(22) unaligned,
        3 project_id           char(9) unaligned,
      2 transaction_count      fixed bin,
      2 no_attendees           fixed bin,
      2 attendee               (user_list_no_attendees
                                 refer (user_list.no_attendees)),
        3 person_id            char(22) unaligned,
        3 project_id           char(11) unaligned,
        3 attending            bit(1) unaligned,
        3 mbz1                 bit(2) unaligned,
        3 notify               bit(1) unaligned,
        3 removed              bit(1) unaligned,
        3 read_only            bit(1) unaligned,
        3 mbz2                 bit(3) unaligned,
        3 last_time_attended   fixed bin(71),
        3 highest_trans_seen   fixed bin;
```

STRUCTURE ELEMENTS

version
is the version number of this structure. Currently, this is set to the constant user_list_version_1 which is also declared in the include file.

person_id
is the Person_id of the chairman of the meeting.

project_id
is the Project_id of the chairman of the meeting.

transaction_count
is the total number of transactions in the meeting.

no_attendees
is the total number of non-deleted participants in the meeting.

person_id
is the Person_id of this participant.

project_id
is the Project_id of this participant.

attending
is set if this participant is currently attending the meeting.

mbz1
is not used.

notify
is set if this participant has requested notifications.

removed
is set if this participant has turned off the participating switch.

read_only
is set if this user may not enter transactions in the meeting.

mbz2
is not used.

last_time_attended
is the time the participant last attended the meeting.

highest_trans_seen
is the index of the most-recently entered transaction that the user has read.

Entry: forum_$list_users_idx

This entry is used to extract information about the participants in a Forum meeting. It is the same as the list_users entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_$list_users_idx entry (fixed bin, ptr, ptr,
      fixed bin(35));

call forum_$list_users_idx (forum_idx, area_ptr, user_list_ptr, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

area_ptr
is a pointer to an area where the user_list structure is to be allocated. (Input) It can not be null.

user_list_ptr
is a pointer to the user_list structure described under the forum_$list_users entry point. (Output).

status
is a standard status code. (Output)

Entry: forum_$open_forum

This entry opens a meeting and returns the forum_idx used in many of the other calls to forum_.

USAGE

```
declare forum_$open_forum entry (char(*), char(*), fixed bin,
      fixed bin(35));

call forum_$open_forum (dirname, entryname, forum_idx, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

forum_idx
is the index number used in subsequent calls to forum_ that take a forum index. All coexistent openings of the same meeting in the same process will return the same number. (Output)

status
is a standard status code. If this is non-zero the meeting could not be opened. (Output) It can have one of the following values:

forum_error_table_$meeting_adjourned
The chairman has adjourned the meeting. Only the chairman can attend.

forum_error_table_$no_such_forum
The dirname-entryname pair does not exist.

forum_error_table_$not_a_forum
The segment named by dirname and entryname is not a Forum meeting.

forum_error_table_$not_eligible
The caller has insufficient access to participate in the meeting.

Entry: forum_$read_trans

This entry is used to read a transaction from a Forum meeting.

USAGE

```
declare forum_$read_trans entry (fixed bin, fixed bin, ptr, ptr,
      fixed bin(35));

call forum_$read_trans (forum_idx, trans_idx, area_ptr,
      forum_user_trans_ptr, status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

trans_idx
: is the index of the transaction to be read. (Input)

area_ptr
: is a pointer to an area where the forum_user_trans structure is to be allocated. (Input)

forum_user_trans_ptr
: is a pointer to the forum_user_trans structure described below. (Output)

status
: is a standard status code. (Output) It can have the following values:

  forum_error_table_$invalid_trans_idx
  : There is no transaction corresponding to the supplied index.

  forum_error_table_$trans_deleted
  : The deleted switch is set for this transaction and the caller is neither the chairman nor the author.

  forum_error_table_$trans_reaped
  : The transaction has been physically removed from the meeting.

NOTES

The forum_user_trans structure is declared in forum_user_trans.incl.pl1 and has the following structure:

```
dcl 1 forum_user_trans          based (forum_user_trans_ptr) aligned,
      2 type                    fixed bin,
      2 person_id               char(22),
      2 project_id              char(9),
      2 time                    fixed bin(71),
      2 trans_no                fixed bin,
      2 next_trans_ptr          ptr,
      2 prev_trans_ptr          ptr,
      2 subject_length          fixed bin,
      2 text_length             fixed bin,
      2 unfilled                bit(1) aligned,
      2 subject                 unaligned char(0 refer
                                forum_user_trans.subject_length))
      2 text                    unaligned char(0 refer
                                forum_user_trans.text_length));
```

STRUCTURE ELEMENTS

type
Currently, this field is unused.

person_id
is the Person_id of the author of this transaction.

project_id
is the Project_id of the author of this transaction.

time
is the clock reading of the time the transaction was entered.

trans_no
is the index of this transaction.

next_trans_ptr
is intended to be used to thread transactions together. It is set to null.

prev_trans_ptr
is intended to be used to thread transactions together. It is set to null.

subject_length
is the length, in characters, of the subject of the transaction.

text_length
is the length, in characters, of the text of the transaction.

unfilled
is set if the transaction should not be filled on output.

subject
is the subject of the transaction.

text
is the text of the transaction.

Entry: forum_$real_forum_limits

This entry is used to obtain various pieces of information about an open Forum meeting.

USAGE

```
declare forum_$real_forum_limits entry (fixed bin, fixed bin, fixed bin,
      fixed bin, fixed bin, fixed bin, bit(36) aligned, fixed bin(35));

call forum_$real_forum_limits (forum_idx, type, last_seen, first_trans,
      last_trans, new_trans_count, flags, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

type
specifies whether the values returned in last_seen, first_trans, and last_trans refer to deleted transactions. The constants ONLY_UNDELETED, INCLUDE_DELETED, and ONLY_DELETED are declared in forum_user_trans.incl.pl1. (Input)

last_seen
is the index of the most-recently entered transaction that the user has seen, as set by forum_$set_last_seen_idx. (Output)

first_trans
is the index of the oldest transaction in the meeting. (Output)

last_trans
is the index of the most recent transaction in the meeting. (Output)

new_trans_count
is the number of transactions entered since last_seen by users other than the caller. (Output)

flags
is a collection of switches as described below. (Output)

status
is a standard status code. (Output)

NOTES

The flags argument is declared in forum_flags.incl.pl1 and has the following strucure:

```
dcl 1 forum_flags              aligned based (addr (forum_flags_word)),
      2 chairman               bit(1) unaligned,
      2 read_only              bit(1) unaligned,
      2 print_cm_message       bit(1) unaligned,
      2 print_acl_message      bit(1) unaligned,
      2 acl_has_changed        bit(1) unaligned,
      2 adjourned              bit(1) unaligned,
      2 mbz                    bit(30) unaligned;
```

STRUCTURE ELEMENTS

chairman
is set if the user is the chairman of the meeting.

read_only
is set if the user may not enter transactions in the meeting.

print_cm_message
is set if the chairman message has changed since the last time the user saw it.

print_acl_message
is set if the access eligibility message should be printed.

acl_has_changed
is set if the access to the meeting has changed since the last time the user was informed that it had changed.

adjourned
is set if the chairman has adjourned the meeting.

mbz
is not used.

Entry: forum_$real_trans_ref_info

This entry is used to obtain information about an individual transaction.

USAGE

```
declare forum_$real_trans_ref_info entry (fixed bin, fixed bin,
      fixed bin, fixed bin, fixed bin, bit(1) aligned, fixed bin(35));

call forum_$real_trans_ref_info (forum_idx, type, trans_idx, prev_ref,
      next_ref, deleted_sw, status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

type
: specifies whether the values returned in prev_ref and next_ref refer to deleted transactions. The constants ONLY_UNDELETED, INCLUDE_DELETED, and ONLY_DELETED are declared in forum_user_trans.incl.pl1. (Input)

trans_idx
: is the index of the transaction for which information is desired. (Input)

prev_ref
: is the index of the preceding transaction in the same transaction chain. This is zero if the transaction is the first in a chain. (Output)

next_ref
: is the index of the next transaction in the same transaction chain. This is zero if the transaction is the last in a chain. (Output)

deleted_sw
: is set if the transaction has been marked as deleted. (Output)

status
: is a standard status code. (Output) It can have one of the following values:

  forum_error_table_$invalid_trans_idx
  : There is no transaction corresponding to the supplied index.

  forum_error_table_$trans_reaped
  : The transaction has been physically removed from the meeting.

Entry: forum_$refuse_notifications

This entry turns off reception of forum notifications by the calling process.

USAGE

```
declare forum_$refuse_notifications entry (fixed bin(35));

call forum_$refuse_notifications (status);
```

ARGUMENTS

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$no_notify_seg
The notifications database could not be initiated.

Entry: forum_$set_delete_sw

This entry is used to delete or retrieve a transaction.

USAGE

```
declare forum_$set_delete_sw entry (fixed bin, fixed bin,
      bit(1) aligned, fixed bin(35));

call forum_$set_delete_sw (forum_idx, trans_idx, delete_sw, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

trans_idx
is the index of the transaction whose delete switch is to be changed. (Input)

delete_sw
should be on to delete the transaction, or off to retrieve it.

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$chairman_only
The caller is neither the author of the transaction nor the chairman of the meeting.

forum_error_table_$invalid_trans_idx
There is no transaction corresponding to the supplied index.

Entry: forum_$set_event_channel

This entry is used to set the event channel used to send wakeups when interesting forum events occur.

USAGE

```
declare forum_$set_event_channel entry (char(*), char(*), fixed bin(71),
      fixed bin(35));

call forum_$set_event_channel entry (dirname, entryname, event_channel,
      status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

event_channel
is the event channel which should receive wakeups from other users attending the same meeting. (Input)

status
is a standard status code. (Output)

NOTES

Currently, the only wakeup sent by Forum occurs when a transaction is entered while you are attending the meeting. The message passed with this wakeup is:

```
declare 1 forum_message,
          2 uid                bit(36) aligned,
          2 attendee_idx       fixed bin;
```

STRUCTURE ELEMENTS

uid
is the unique identifier of the meeting. It should be passed to forum_$validate_uid to ensure that the user is still attending that meeting.

attendee_idx
identifies the user who entered the transaction. A character string containing the name of that user can be obtained by passing this to forum_$convert_attendee_idx.

Entry: forum_$set_event_channel_idx

This entry is used to set the event channel used to send wakeups when interesting forum events occur.

USAGE

```
declare forum_$set_event_channel_idx entry (fixed bin, fixed bin(71),
      fixed bin(35));

call forum_$set_event_channel_idx entry (forum_idx, event_channel,
      status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

event_channel
: is the event channel which should receive wakeups from other users attending the same meeting. (Input)

status
: is a standard status code. (Output)

NOTES

See the forum_$set_event_channel entry point for a description of interprocess messages used by Forum.

Entry: forum_$set_last_seen_idx

This entry is used to set the last_seen_idx for a user. This index denotes the highest transaction that the user has read.

USAGE

```
declare forum_$set_last_seen_idx (fixed bin, fixed bin, bit(1) aligned,
      fixed bin(35));

call forum_$set_last_seen_idx (forum_idx, last_seen_trans_idx,
      force_switch, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

last_seen_trans_idx
is the index of the transaction to be marked as the most recent seen by this user. (Input)

force_switch
If this switch is on, the last_seen_idx for this user is always set to last_seen_trans_idx. If it is off, the maximum of last_seen_trans_idx and the current last_seen_idx for this user is used. (Input)

status
is a standard status code. (Output)

Entry: forum_$set_message

This entry is used to set a chairman's greeting message for a Forum meeting.

USAGE

```
declare forum_$set_message entry (fixed bin, char(*), fixed bin(35));

call forum_$set_message (forum_idx, message, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

message
is the text message, this can contain at most 256 characters. (Input)

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$chairman_only
The caller is not the chairman of the meeting.

forum_error_table_$message_too_long
The message exceeds the 256 character limit.

Entry: forum_$set_switch

This entry is used to change several switches associated with a Forum meeting or its participants.

USAGE

```
declare forum_$set_switch entry (char(*), char(*), char(*), char(*),
      bit(1) aligned, fixed bin(35));

call forum_$set_switch (dirname, entryname, person_id, switch_name,
      switch_setting, status);
```

ARGUMENTS

dirname
: is the pathname of the directory containing the meeting. (Input)

entryname
: is the name of the meeting. The ".control" suffix is required. (Input)

person_id
: is the Person_id of the user on whose behalf you are setting the switch. The Project_id should not be given. Starnames are not permitted. If this is blank, the current user is assumed. Only the chairman of a forum can set switches for persons other than herself. (Input)

switch_name
: is the name of the switch to set. See "List of Switches" below. (Input)

switch_setting
: is the new switch setting. (Input)

status
: is a standard status code. It can have one of the following values:

  forum_error_table_$chairman_only
  : Person_id does not identify the caller and the caller is not the chairman.

  forum_error_table_$invalid_switch_name
  : The switch_name is not one of the valid switches.

  forum_error_table_$switch_not_changed
  : The old setting was the same as the new setting.

LIST OF SWITCHES

adjourned, adj
: Prevents users from entering the meeting. Only the chairman can set this switch.

meeting_eligibility_messages, mtg_emsg
Turns on printing of eligibilty messages for this meeting. This switch can only be set by the chairman.

notify, nt
Turns on the notify switch for the meeting. This means that an interactive message will be sent each time a transaction is entered if the user is logged in and has issued the forum_accept_notifications command.

participating, part
Turns on the participation switch for the meeting.

Entry: forum_$set_switch_idx

This entry is used to change several switches associated with a Forum meeting or its participants. It is the same as the set_switch entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_$set_switch_idx entry (fixed bin, char(*), char(*),
      bit(1) aligned, fixed bin(35));

call forum_$set_switch_idx (forum_idx, person_id, switch_name,
      switch_setting, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

person_id
is the Person_id of the user on whose behalf you are setting the switch. The Project_id should not be given. Starnames are not permitted. If this is blank, the current user is assumed. Only the chairman of a forum can set switches for persons other than herself. (Input)

switch_name
is the name of the switch to set. See "List of Switches" below. (Input)

switch_setting
is the new switch setting. (Input)

status
is a standard status code. It can have one of the following values:

forum_error_table_$chairman_only
Person_id does not identify the caller and the caller is not the chairman.

forum_error_table_$invalid_switch_name
The switch_name is not one of the valid switches.

forum_error_table_$switch_not_changed
The old setting was the same as the new setting.

LIST OF SWITCHES

access_changed
Specifies that the ACL on the meeting has changed. This switch is set by the forum_chairman_$set_forum_acl entry point, and should normally only be turned off by this subroutine.

adjourned, adj
Prevents users from entering the meeting. Only the chairman can set this switch.

deleted
Specifies that the attendee record for this user should be marked as deleted.

meeting_eligibility_messages, mtg_emsg
Turns on printing of eligibilty messages for this meeting. This switch can only be set by the chairman.

message_seen
Specifies that the chairman message on the meeting has changed. This switch is set by the forum_$set_message entry point, and should normally only be turned off by this subroutine.

notify, nt
Turns on the notify switch for the meeting. This means that an interactive message will be sent each time a transaction is entered if the user is logged in and has issued the forum_accept_notifications command.

participating, part
Turns on the participation switch for the meeting.

Entry: forum_$trans_ref_info

This entry is used to obtain information about an individual transaction. This entry is obsolete. The forum_$real_trans_ref_info entry should be used instead.

USAGE

```
declare forum_$trans_ref_info entry (fixed bin, fixed bin, fixed bin,
      fixed bin, bit(1) aligned, fixed bin(35));

call forum_$trans_ref_info (forum_idx, trans_idx, prev_ref, next_ref,
      deleted_sw, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

trans_idx
is the index of the transaction for which information is desired. (Input)

prev_ref
is the index of the preceding transaction in the same transaction chain. This is zero if the transaction is the first in a chain. (Output)

next_ref
is the index of the next transaction in the same transaction chain. This is zero if the transaction is the last in a chain. (Output)

deleted_sw
is set if the transaction has been marked as deleted. (Output)

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$invalid_trans_idx
There is no transaction corresponding to the supplied index.

forum_error_table_$trans_reaped
The transaction has been physically removed from the meeting.

Entry: forum_$trans_time_info

This entry returns the indexes of transactions entered during a given time span.

USAGE

```
declare forum_$trans_time_info entry (fixed bin, fixed bin(71),
      fixed bin(71), fixed bin, fixed bin, fixed bin(35));

call forum_$trans_time_info (forum_idx, low_time, high_time, low_trans,
      high_trans, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

low_time
is the timestamp of the low end of the time span. (Input)

high_time
is the timestamp of the high end of the time span. (Input)

low_trans
is the index of the first transaction entered after low_time. If there are no transactions that new, than the index of the highest transaction in the meeting is returned. (Output)

high_trans
is the index of the transaction entered immediately before high_time. If no transactions were entered before then, zero is returned. (Output)

status
is a standard status code. (Output).

Entry: forum_$validate_uid

This entry is used to determine if a given meeting unique identifier (uid) corresponds to a given meeting.

USAGE

```
declare forum_$validate_uid entry (fixed bin, bit(36) aligned,
      fixed bin(35));

call forum_$validate_uid (forum_idx, forum_uid, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

forum_uid
is the uid of the meeting. Normally this is obtained from Forum interprocess messages. (Input)

status
is a standard status code. (Output) It can have the following value:

forum_error_table_$incorrect_uid
The supplied uid is not the unique identifier of the meeting identified by forum_idx.

Name: forum_chairman_

The forum_chairman_ subroutine provides entries to perform chairman duties on Forum meetings.

Entry: forum_chairman_$change_chairman

This entry is used to change the chairman of a Forum meeting.

USAGE

```
declare forum_chairman_$change_chairman entry (char(*), char(*),
      char(*), fixed bin(35));

call forum_chairman_$change_chairman (dirname, entryname, chairman,
      status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

chairman
is the name of the new chairman of the meeting, in the form Person.Project. The new chairman must already be a participant. (Input)

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$chairman_only
The caller is not the current chairman.

forum_error_table_$no_such_user
The supplied new chairman is not a participant.

Entry: forum_chairman_$change_chairman_idx

This entry changes the chairman of a Forum meeting. It is the same as the change_chairman entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_chairman_$change_chairman_idx entry (fixed bin, char(*),
      fixed bin(35));
```

```
call forum_chairman_$change_chairman_idx (forum_idx, chairman, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

chairman
is the name of the new chairman of the meeting, in the form Person.Project. The new chairman must already be a participant. (Input)

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$chairman_only
The caller is not the current chairman.

forum_error_table_$no_such_user
The supplied new chairman is not a participant.

Entry: forum_chairman_$chname_forum

This entry is used to change the names of Forum meetings.

USAGE

```
declare forum_chairman_$chname_forum entry (char(*), char(*), char(*),
      char(*), fixed bin(35));

call forum_chairman_$chname_forum (dirname, entryname, old_name,
      new_name, status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

old_name
is a name to be removed from the meeting, or blank if no name is to be removed. If non-blank, the ".control" suffix is required. (Input)

new_name
is a name to be added to the meeting, or blank if no name is to be added. If non-blank, the ".control" suffix is required. (Input)

status
is a standard status code. (Output)

Entry: forum_chairman_$chname_forum_idx

This entry is used to change the names of Forum meetings. It is the same as the chname_forum entry point except that a forum_idx is used to identify the meeting instead of a pathname.

USAGE

```
declare forum_chairman_$chname_forum_idx entry (fixed bin, char(*),
      char(*), fixed bin(35));

call forum_chairman_$chname_forum_idx (forum_idx, old_name, new_name,
      status);
```

ARGUMENTS

forum_idx
: identifies a meeting opened by forum_$open_forum. (Input)

old_name
: is a name to be removed from the meeting, or blank if no name is to be removed. If non-blank, the ".control" suffix is required. (Input)

new_name
: is a name to be added to the meeting, or blank if no name is to be added. If non-blank, the ".control" suffix is required. (Input)

status
: is a standard status code. (Output)

Entry: forum_chairman_$create_forum

Creates a Forum meeting with the user as chairman.

USAGE

```
declare forum_chairman_$create_forum entry (char(*), char(*),
      fixed bin(35));

call forum_chairman_$create_forum (dirname, entryname, status);
```

ARGUMENTS

dirname
    is the pathname of the directory containing the meeting. (Input)

entryname
    is the name of the meeting. The ".control" suffix is required. (Input)

status
    is a standard status code.

ACCESS REQUIRED

The caller must have modify permission on the containing directory.

Entry: forum_chairman_$expunge

This entry is used to physically remove deleted participant records and deleted transactions from a Forum meeting. This results in reclaimed storage space.

USAGE

```
declare forum_chairman_$expunge entry (fixed bin, bit (36) aligned,
      fixed bin, fixed bin, fixed bin, fixed bin, fixed bin(35));

call forum_chairman_$expunge (forum_idx, switches, users_deleted,
      trans_deleted, users_damaged, trans_damaged, status);
```

ARGUMENTS

forum_idx
    identifies a meeting opened by forum_$open_forum. (Input)

switches
is a word of switches as follows: (Input)

transactions
if set, deleted transactions should be expunged.

users
if set, deleted participant records should be expunged.

users_deleted
is the number of participant records which were expunged. (Output)

trans_deleted
is the number of transactions which were expunged. (Output)

users_damaged
is the number of participant records which were expunged because they were inconsistent. (Output)

trans_damaged
is the number of transactions which were expunged because they were inconsistent. (Output)

status
is a standard status code. (Output)

Entry: forum_chairman_$set_forum_acl

This entry is used to add or change an ACL term on a Forum meeting. ACL terms can only be specified for Person_id.* or *.Project_id, terms of the form Person_id.Project_id are not allowed.

USAGE

```
declare forum_chairman_$set_forum_acl entry (fixed bin, char(*),
      bit(1) aligned, bit(1) aligned, bit(1) aligned, fixed bin(35));

call forum_chairman_$set_forum_acl (forum_idx, access_name, user_sw,
      set_sw, write_sw, status);
```

ARGUMENTS

forum_idx
identifies a meeting opened by forum_$open_forum. (Input)

access_name
The Person_id or Project_id whose ACL is to be added or deleted. Access_name can not contain a ".". (Input)

forum_chairman_

user_sw
: This switch should be on if access_name is a Person_id, and off if it is a Project_id. (Input)

set_sw
: This switch should on if access is to be added, and off if it is to be deleted. (Input)

write_sw
: This switch should on if write access is to be added, and off if only read access is to be added. This switch is ignored if set_sw is off. (Input)

status
: is a standard status code. (Output)

ACCESS REQUIRED

The caller must have modify permission on the containing directory.

Name: forum_admin_

The forum_admin_ subroutine provides a site administrative interface to the Forum Interactive Meeting System.

Entry: forum_admin_$change_chairman

This entry is used to change the chairman of a Forum meeting by a system administrator who is not the current chairman of the meeting.

USAGE

```
declare forum_admin_$change_chairman entry (char(*), char(*), char(*),
      fixed bin(35));

call forum_admin_$change_chairman (dirname, entryname, chairman,
      status);
```

ARGUMENTS

dirname
is the pathname of the directory containing the meeting. (Input)

entryname
is the entryname of the meeting. (Input) The ".control" suffix is required.

chairman
is the name of the new chairman of the meeting, in the form Person_id.Project_id. The new chairman must already be a participant. (Input)

status
is a standard status code. (Output) It can have one of the following values:

forum_error_table_$no_such_user
The supplied new chairman is not a participant.

ACCESS REQUIRED

The caller must have modify permission on the containing directory.

Entry: forum_admin_$delete_forum

This entry is used to delete a meeting by a system administrator who is not the current chairman of the meeting.

USAGE

```
declare forum_admin_$delete_forum entry (char(*), char(*),
      fixed bin(35));

call forum_admin_$delete_forum (dirname, entryname, status);
```

ARGUMENTS

dirname
    is the pathname of the directory containing the meeting. (Input)

entryname
    is the entryname of the meeting. The ".control" suffix is required. (Input)

status
    is a standard status code. (Output)

ACCESS REQUIRED

The caller must have modify permission on the containing directory.

Entry: forum_admin_$init_notifications

This entry is used to create the notifications database.

USAGE

```
declare forum_admin_$init_notifications entry (fixed bin(35));

call forum_admin_$init_notifications (status);
```

ARGUMENTS

status
    is a standard status code. (Output)

ACCESS REQUIRED

The caller must have "sma" access to the central forum directory.

Entry: forum_admin_$set_forum_acl

This entry is used to add or change an ACL term on a meeting. ACL terms can only be specified for Person_id.* or *.Project_id, terms of the form Person_id.Project_id are not allowed. The caller need not be the chairman of the meeting.

USAGE

```
declare forum_admin_$set_forum_acl entry (char(*), char(*), char(*),
      bit(1) aligned, bit(1) aligned, bit(1) aligned, fixed bin(35));

call forum_admin_$set_forum_acl (dirname, entryname, access_name,
      user_sw, set_sw, write_sw, status);
```

ARGUMENTS

dirname
: is the pathname of the directory containing the meeting. (Input)

entryname
: is the entryname of the meeting. The ".control" suffix is required. (Input)

access_name
: is the Person_id or Project_id whose ACL is to be added or deleted. Access_name can not contain a ".". (Input)

user_sw
: This switch should be on if access_name is a Person_id, and off if it is a Project_id. (Input)

set_sw
: This switch should be on if access is to be added, and off if it is to be deleted. (Input)

write_sw
: This switch should on if write access is to be added, and off if only read access is to be added. This switch is ignored if set_sw is off. (Input)

status
: is a standard status code. (Output)

ACCESS REQUIRED

The caller must have modify permission on the containing directory.

Entry: forum_admin_$set_global_switch

This entry is used to set switches associated with the forum environment for an entire site.

USAGE

```
declare forum_admin_$set_global_switch entry (char(*), bit(1) aligned,
        fixed bin(35));

call forum_admin_$set_global_switch (switch_name, value, status);
```

ARGUMENTS

switch_name
: is the name of the switch to be set. It can be one of "print_eligiblity_messages", "pemsg", "chairman_set_eligibilty_msg", or "cm_set_emsg". (Input)

value
: is the value to be assigned to the switch. (Input)

status
: is a standard status code. (Output) It can have the following value:

  forum_error_table_$invalid_switch_name
  : The switch_name given was not one of those listed above.

ACCESS REQUIRED

The user must have "rw" access to forum_data_.

Entry: forum_admin_$set_switch

This entry is used to change several switches associated with a Forum meeting or its participants. The caller need not be the chairman of the meeting in order to set any of the switches.

USAGE

```
declare forum_admin_$set_switch entry (char(*), char(*), char(*),
        char(*), bit(1) aligned, fixed bin(35));

call forum_admin_$set_switch (dirname, entryname, person_id,
        switch_name, switch_setting, status);
```

ARGUMENTS

dirname
: is the pathname of the directory containing the meeting. (Input)

entryname
is the name of the meeting. The ".control" suffix is required. (Input)

person_id
is the Person_id of the user on whose behalf you are setting the switch. The Project_id should not be given. Starnames are not permitted. If this is blank, the caller's Person_id is used. (Input)

switch_name
is the name of the switch to set. See "List of Switches" below. (Input)

switch_setting
is the new switch setting. (Input)

status
is a standard status code. It can have one of the following values:

forum_error_table_$invalid_switch_name
the switch_name is not one of the valid switches.

forum_error_table_$switch_not_changed
the old setting was the same as the new setting.

LIST OF SWITCHES

access_changed
Specifies that the ACL on the meeting has changed. This switch is set by the forum_chairman_$set_forum_acl entrypoint, and should normally only be turned off by this subroutine.

adjourned, adj
Prevents users from entering the meeting.

deleted
Specifies that the attendee record for this user should be marked as deleted.

meeting_eligibility_messages, mtg_emsg
Turns on printing of eligibilty messages for this meeting.

message_seen
Specifies that the chairman message on the meeting has changed. This switch is set by the forum_$set_message entrypoint, and should normally only be turned off by this subroutine.

notify, nt
Turns on the notify switch for the meeting. This means that an interactive message will be sent each time a transaction is entered if the user is logged in and has issued the forum_accept_notifications command.

participating, part
Turns on the participation switch for the meeting.

forum_admin_

ACCESS REQUIRED

The caller must have either modify permission on the containing directory or write access to the ".control" segment.

APPENDIX C

GLOSSARY

Like every system, Forum has some terms that have special meaning. This section will attempt to make some of them clear.

attending, attendee
A user is attending a meeting if he or she is currently using the Forum system and has entered that meeting (using the forum "goto" request). Attending is analogous to being inside the room where the meeting is being held, as opposed to participating, which is like having been there and gone out for coffee.

chairman
The chairman of a meeting is the user who controls the activities in the meeting. Usually it is the user who created or convened the meeting in the first place, but since the chairmanship can be passed on to another user, this is not always the case. It is the chairman who can adjourn meetings and controls such things as who is eligible to participate in the meeting. He can also delete and retrieve transactions.

control arguments
Forum commands and requests all perform a specific function, and most have special options that modify the default action or tell Forum exactly which transactions or meetings to operate on. Control arguments always start with the "-" character.

current meeting
The current meeting is the one you are currently attending, and remains so until you explicitly attend a different meeting (using the "goto" request) or exit Forum (using the "quit" request).

current transaction
The current transaction is the one that the user has just listed, printed, entered, or otherwise referred to. Since Forum keeps track of the current transaction, it performs requests that are issued with no transaction specifier on the current transaction (this is the default).

eligible
A user is eligible to participate in a meeting if the chairman of the meeting has set the access on the meeting in such a way as to allow that user to participate in the meeting. A user must be eligible to participate before he or she can attend.

flag

Flags are printed when you list either meetings or participants in a meeting. Flags consist of one letter and indicate the user's state with respect to that meeting (e.g., "e" means eligible to participate and "p" means you are already a participant).

Forum

Forum is the system itself and the programs or software that comprise it. Forum enables eligible users to create, enter, participate in, and manage meetings.

forum

When you type "forum" at command level, you are typing the forum command, which enters you into the Forum subsystem (see also Forum).

keyword

A keyword is a transaction specifier consisting of a generic term that describes one or more transactions (e.g., "first" and "all").

meeting

A meeting consists of a chairman, a list of participants, and the proceedings or minutes. Meetings are Forum's reason for existence. There will likely be many meetings on a system, each created to discuss some specific or general topic or issue. Each meeting is identified by a long, descriptive name (such as Everybody's_Meeting), and at least one short, easy-to-use name (such as eve).

participate, participant

A user is considered to be participating in a meeting if he or she has entered that meeting at least once. The first time a user goes to a meeting, a record of that user's participation in that meeting is created, making the user an official participant.

proceedings

The proceedings of a meeting are the record maintained by Forum of what exactly has been said during the life of the meeting, and by whom. This record of what took place can be examined in a flexible fashion by participants of the meeting.

regular expression

Regular expressions are transaction specifiers used with Forum requests. They consist of character strings (letters, numbers, words, or phrases) enclosed in slashes (/) to be matched against the text of transactions. Certain special characters can be used for fancier matching techniques. See Section 4.

request

A request is how a participant tells Forum to perform some action, such as entering a meeting, printing out transactions, or finding out the names of the other participants in a meeting. Forum requests are very similar in syntax to Multics commands.

request line

A request line is a Forum request, optionally followed by arguments, control arguments, and transaction specifiers.

request loop

The request loop is the part of Forum that reads the request you type, performs the specified operation, and finishes with a prompt to you (forum:) for another request. This is a repeating cycle until you exit Forum.

search list

The forum search list is a list of pathnames of directories that contain meetings which you want to attend. It is manipulated via the add_search_paths and delete_search_paths commands, and printed with the print_search_paths command (all described in the Commands manual).

search path

A search path is a pathname of a directory in your search list.

transaction

A transaction is a statement made by an attendee in a meeting. Entering a transaction is Forum's equivalent of speaking at a real meeting. The text of the transaction, its subject, the date and time it was entered, and the name of the participant who entered it are placed in the proceedings of a meeting. Other participants can then use Forum requests to manipulate the transactions of the meeting, such as "print" and "list" to look at the transaction.

transaction chain

A transaction chain is a series of transactions that are related in that they all refer to one previous transaction. In other words, it is a discussion centered on the transaction of one user, linked together by a common subject (see "Transaction Chains" above).

transaction specifier

A transaction specifier is a way of choosing a transaction or a group of transactions for Forum to list, print, write, or otherwise refer to. Transaction specifiers come in many forms: transaction numbers, keywords, control arguments, and regular expressions.

unprocessed transaction

An unprocessed transaction is a transaction whose text you have prepared either in response to another transaction, or to start a new discussion in a meeting, but have not yet entered it into the proceedings of the meeting. It can be manipulated in a number of ways (see Section 4).

INDEX

F

S

T