MUSIC/SP Administrator's Reference

Part 3 - Chapters 10 to 16 (AR_P3.PS)

# Chapter 10. TMENU - Tailoring the User View

## Overview of TMENU

TMENU is the name of the program on MUSIC for creating tailored user views of MUSIC. This facility is available for making your own menus, or tailoring existing menus to better suit your needs and those of your users. Some of the available features include filtering the commands to which users of a particular view have access, signing off from MUSIC when the menu is terminated, and executing your own programs. A sample MUSIC view for programmers, called PROG, can be modified to suit your installation's needs. Programs that are used often can be added to the menu or items on the menu can be exchanged for other programs offered at your installation. For example, figure 10.1 which lists PROG.MENU, the companion menu for PROG, can be added to or altered so that when the sample program PROG is executed, the facilities you defined are available to the users of this view.

## Making a Menu

The menu is used to provide four functions:

1. provides TMENU with the menu items and descriptive text.
2. provides TMENU with the names of program files to be executed as required by the user.
3. provides labels for the menu items for menu selection.
4. provides for default parameters to be passed to the respective program.

```
)+ - PROG
)% -  PROGRAMMER'S MENU  -
1. EDIT     - edit an existing file
2. EDIT     - edit a new file
3. LIBRARY  - look at your library
4. PASSWORD - change your sign-on password
5. PROFILE  - change your code options
6. NEWS     - list latest news
)1 EDITOR
)2 EDITOR-NEW
)3 TODO.LIB
)4 PROFILE>NEWPW
)5 PROFILE
)6 NEWS/
```

*Figure 10.1 - Listing of PROG.MENU*

## The General Format of the Menu

The menu is made up of four types of specification lines and is divided into two logical components. The first component describes the menu items (selection codes) and the descriptive text. The second component describes the actual program names and their parameters if any. The program names and the menu items are connected by the *option code*. For example, item 'H' in Figure 10.2 has both a descriptive entry in component one as well as a program name and parameters definition in component two.

```
)+                            <--------  Menu start flag
)% -  TITLE OF MENU  -        <--------  Menu title line

1. Item One                   <--------
2. Item Two                           |  Option codes
P. Profile, print $ remaining         |  and descriptions
H. Help                       <--------

)1 PROG1                      <--------  Corresponding
)2 PROG2                              |  file and
)P PROFILE/PRINT$                     |  parameter
)H HELP                       <--------  specifications
```

*Figure 10.2 - Sample Menu for User View Tailoring*

)+ is the marker that indicates that the file being processed is a menu file. It must be the first line of a menu file and appear only once. An optional 1-16 character menu name can be placed on this line. This text will be placed on the extreme right of the title line of the menu.

)% is the title marker. The title marker provides a title for the menu posted on the screen. A blank must follow the "%" character. If more than one title line is present, the last one will be used.

The title itself will be centered by the menu formatter of TMENU, therefore do not attempt to center it. The standard that should be followed is to place a dash (-) before and after the title text separated by one or two blanks. Example:

)% - title -

The title will be centered and padded by dashes.

---------- title ----------

xx. this is the one or two character combination that the user will know the item by. This becomes the item option code or name.

The option code character(s) must be directly followed by a period and a blank (. ).

## General Format of the Option Specification Line.

```
Type 1

   xx. OPTION  -  Description
```

| | |
|---|---|
| xx. | is the 1 or 2 character option code followed by a period and a blank. |
| OPTION | is the option name (usually in upper case). |
| - | is an optional delimiter. |
| Description | is the option description. |

```
Type 2

   xx. Description
```

| | |
|---|---|
| xx. | is the 1 or 2 character option code followed by a period and a blank. |
| Description | is the option description. |

# Guidelines for Specifying Option Lines

## Option Code

- The option code can be numbers or characters except period and close parenthesis.
- Do not use roman numerals, arabic numerals are more familiar and more easily distinguishable. When a short number of items are used letters indicative of the function performed by the item is appropriate, for example, 'P' for Profile and 'H' for Help.
- If letters and numerals are to be mixed, it is best to place all the numeric option lines in sequential order followed by the letter option codes.
- The option lines should be in the order of greatest anticipated usage (not withstanding the previous point). This will facilitate usage.

## Option Name

- The option name is not always used or appropriate, however when this format is used it should be in upper case. (Type 1)

## Option Description

- The option description should begin with a capital letter.
- The description should be a statement with correct syntax. Statements that seem to be questions should

be avoided.

```
        1. Submit a File
        11. Submit a File
        H. Help on File
        1. HELP  -  Get help on file usage
```

*Figure 10.3 - Sample Option Specification Menu*

## Program Specification Lines

Besides the program name and the default parameters (program options), two other types of information can be included on the program specification line. The first is whether the user's parameters are to override, not override, or be appended to the default parameters. The second is whether the program is to be an 'always' program, a noncancelable program, or just an ordinary program (that is neither of the first two characteristics).

```
  )xx programYparametersZ
```

| | |
|---|---|
| )xx | The required close parenthesis indicates a program specification line. *xx* is a 1- 2-character item name or option code. There should be as many of these lines as there were item specification lines. These lines define the files to be executed and the default parameters if any. |
| program | is the file name to be scheduled for execution. |
| Y | is one of 4 optional delimiters (\| + - >). |
| parameters | is the optional parameter list. |
| Z | is one of 7 optional program types (blank % ! " ? / <). |

## Defining Parameter Processing and Passing

The parameters are separated from the program name by one of four special characters. These characters are shown in Figure 10.4.

```
      |  vertical bar   - user if specified, otherwise
                          the default parm
      +  plus sign      - user parm prefixed to defaults
      -  minus sign     - user parm appended to default
      >  greater than   - user parm ignored
```

*Figure 10.4 - Parameter Processing Flags for User View Tailoring*

Each of these serves to delimit the program name from the parameters as well as to indicate the parameter

handling option.

The following defines and describes these four delimiter characters.

| The vertical bar indicates that any user specified parameters are to override the default parameters. That is either the user's parameters or the default parameters are used and not both.

+ The plus sign indicates that any user specified parameters are not to override the default parameters. Rather the user parameters are to be prefixed to the default parameters. That is both specifications are passed. The two parameter lists will be separated by a blank.

- The minus sign indicates that any user specified parameters are not to override the default parameters. Rather the user parameters are to be appended to the default parameters. That is both specifications are passed. The two parameter lists will be separated by a blank.

> The greater than sign indicates that any user specified parameters are to be ignored. Only the default parameters are passed.

## Defining Program Types

One of five characters can be used to describe the program type. These characters and their functions are given in Figure 10.5.

```
" " no type flag     -  "nonalways"/cancellable
 %  percent sign     -  "always"/noncancellable
 !  exclamation mark -  "nonalways"/noncancellable
 "  double quote     -  "always"/cancellable
 ?  question mark    -  menu file
 /  slash           -  MUSIC command
 <  less than sign  -  TMENU command
```

*Figure 10.5 - Program Type Flags for User View Tailoring*

Any one of these can appear as the last character of the parameter list to indicate the program type. These program characters, when specified, must be appended to the parameter list on the program specification line. It will be converted to a blank and is not considered as part of the parameter list. If no parameter list or *parameter option* character (delimiter) is required or used the *program type* character can be appended to the program name.

The scan for the program type is done from right to left. Therefore the *program type* characters can appear as part of the parameter list provided that a program type character is used at the end of the parameter list.

The following defines and describes these five characters.

" " A blank or no program specification indicates that a program is to be a cancellable nonalways program. When no program type character is specified " " is the default.

% The percent sign indicates that a program is to be a noncancellable always program.

! The exclamation mark indicates that a program is to be noncancellable and nonalways.

" The double quote mark indicates that a program is to have the *always* attribute and is to be cancellable.

In brief an *always* program is one that is automatically reinvoked after it has completed running. As well if it scheduled a program via a NXTPGM call, at the completion of the scheduled program the always program is reinvoked.

? The question mark indicates that the program name is really a menu file.

/ The slash indicates that this is a MUSIC command string. Any commands listed with filter are not allowed here. See the TMENU FILTER option.

< The less than sign indicates that this is a TMENU command.

*Note:* Usage of the program types ", !, and % dictates that the program filenames given with these specifications are limited to 22 character filenames. All other program filenames, menus, or TMENU commands given abide by the normal MUSIC system rules and limits.

```
)3 GORK PARMS! <-default parms are overridable, the program
                  is noncancellable and nonalways.

)3 GORK!       <-default parms are overridable, the program
                  GORK is noncancellable and nonalways.

)3 GORK+PARMS% <-user parms will precede default parms, the
                  program is cancellable and always.

)3 GORK-PARMS  <-user parms will follow default parms, the
                  program is cancellable and nonalways.

)3 GORK>PARMS" <-no user parms, default parms only and gork
                  is an "always" and noncancellable program.

)3 GORK>PARMS  <-no user parms, default parms only, gork is
                  cancellable and nonalways.

)3 GORK?       <-the file name is a menu to be displayed,
                  and not a program.

)3 NEWS/       <-This indicates that news is a MUSIC command
                  and will be executed as such.

)3 END<        <-indicates that the user will be returned to
                  *Go or the calling program.
```

*Figure 10.6 - Sample Program Specification Menu*

# Invoking TMENU

After a menu has been created you must set up an exec file for it. For example, the following is the exec file required for PROG:

```
/INC TMENU
MYNAME='PROG',FMENU='PROG.MENU',STACK='@PROG.STACK'
```

## TMENU Parameters

ACCESS='filename'

Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

**Example:**

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CAL=t or f

Specify either t or f to indicate true or false respectively. If CAL=t is used a calendar is posted on the right of the menu. If CAL=f is used no calendar is displayed. When CAL is not specified the default is CAL=t.

CANCEL=t or f

When CANCEL=t is specified the user can return to *Go by PA1, /CANCEL ALL, or successive END commands or PF3's. However when CANCEL=f, /CANCEL is rejected and normal termination of TMENU results in a sign-off. In other words CANCEL=f prevents users from returning to *Go. The default is CANCEL=t.

ENDCMD='filename'

Where filename is the name of the file to be executed when the user exits the facility. ENDCMD is honoured only when CANCEL=t. filename could be an accounting program that records usage of the facility.

ERRFIL='filename'

Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:TMENU.MSGS[@]. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file has National Language variants E (Spanish), F (French), P (Portuguese), K (Kanji-Japanese) supported through the [@] placeholder in the file name. These files are produced from $TDO:TMENU.MSGS[@].S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:TMENU.MSGS[@].S files are normally off-line and may have to be restored from the source tapes if use is required.

FILTER=t or f

When FILTER=f, any command string which is not an option code or a command used by TMENU is passed to the MUSIC command processor. When FILTER=t, the option codes, the commands used by TMENU, and only the commands and/or programs listed below the parameter line are allowed. The default is FILTER=f.

A number of commands processed by the MUSIC terminal scanner can not be issued from TMENU. These commands are: /COMPRESS, /CTL, /DISCON, /NS, /PAUSE, /PROMPT, /REQUEST, /RUN, /STATUS, /TIME, /TEXT, /TABIN, /TABOUT, /USERS, /VIP, and /WINDOW.

The commands can be entered with any abbreviation up to the minimum abbreviation as specified by an integer to its right. If no minimum abbreviation is specified the command will have to be entered in its entirety as specified. Before the command string is passed the abbreviated name is expanded to its full length. For example, the following abbreviations would be allowed for this list of commands:

| Command | Abbreviations |
|---|---|
| PURGE 2 | purg, pur, pu |
| OUTPUT 3 | outpu, outp, out |
| SUBMIT 2 | submi, subm, sub, su |
| PRINT 3 | prin, pri |

If no commands and/or program file names are specified, then the "/" command is not allowed at all. This is the method for barring the use of the "/" command. For example, in the file ABC:

```
/INC TMENU
MYNAME='ABC',FILTER=T
```

ABC will not allow the use of the "/" command and will use the default menu of TODO.MENU as a first menu. The calendar will be posted on each menu screen. The stack file will be @TMENU.STACK.tcb where *tcb* is the current TCB number.

FMENU=menu — Where *menu* is the name of the first menu file to be displayed. When no menu file is specified the default menu, TODO.MENU[@] is used. This file has National Language variants E (Spanish), F (French), P (Portuguese), K (Kanji-Japanese) supported through the [@] placeholder in the file name.

FS=t or f — When FS=t and TMENU is invoked on a 3270-type terminal, automatic full screen support is provided. On a non full screen terminal such support is never allowed.

When FS=f, no matter what terminal type you are using, only non full screen support will be available. The default is FS=t.

KEYFIL='filename' — Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

Users can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:TMENU.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:TMENU.KEYS have the same format as the lines in the KEYFIL filename.

LOG=t or f — The LOG=t parameter can be used if you wish to keep a usage log. LOG=f is the default.

LOGFIL='filename' — Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@TMENULOG.tcb, where *tcb* is the

current TCB number.

MDELAY=n — Where *n* is a value indicating the delay between attempts at finding out if mail is waiting. When mail is detected, a mail waiting message is posted. When *n* is a positive integer it is taken as representing minutes. When *n* is negative it is taken as seconds. To stop all mail checking, use a value for *n* that is greater than 24 hours such as -86400 secs or 1440 mins. This will prevent the posting of even the initial mail waiting message at the start of the program. The default is MDELAY=-300 or MDELAY=5 (five minutes).

MYNAME='filename' — Where *filename* is the name of the file. In our example it would be PROG. If this parameter does not specify the file name in which it is stored TMENU will not make the file an *always* program.

REMS=t or f — When REMS=t, TMENU will automatically display any reminders for the current day. You can set reminders and query for the day's reminders with the REMIND command or the use of PF2. The default is REMS=t.

When REMS=f the REMIND facility will not be available, and the REMIND command and PF2 will be ignored. The default is REMS=t.

STACK='filename' — Where *filename* is a 1-46 character name of a file to be used as the stack file. The tcb number is appended to it as follows: *filename.tcb*. The stack file is used to store a trace of menus and programs that have been called. In this way the program can return to the correct menu as it drops back through the stack. When no stack file is specified the default "@TMENU.STACK.tcb" is used, where *tcb* is the TCB number.

*Note:* Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

## Built-in Functions

| Command | Description |
|---|---|
| CANCEL | exit the current menu. |
| END | exit the current menu. |
| EXEC | pass the string to MUSIC as a command (abbr. /). See the FILTER option. |
| GETREM | post message reminders for today or no reminders. |
| HELP | provide help on how to manipulate the menu screen. |
| KEYS | post the KEYS screen to enter definitions for the PF keys. |
| MAIL | post message mail waiting or not waiting. |
| OFF | terminate the program and sign off. |
| PFnn | show the definition of PFnn, where *nn* is a number from 1 to 24. |
| PFnn x | set the definition of PFnn to *x*, where *nn* is a number from 1 to 24. The definition can be from 1 to 50 characters long. |
| =x | return to the first screen and process *x*. |
| REMIND | view reminder file (abbr. REM). |
| RETRIEVE | display the previous command in the command area. Up to 5 commands can be recalled. |
| * | echo the previous command. |
| *-n | Display the previous nth command entered in the selection area, where *n* is a number from 0 to 4. |

## Function Key Definitions

| Key | Definition |
|---|---|
| F1/13 | provide help on how to manipulate the menu screen. |
| F2/14 | check today's reminders. |
| F3/15 | terminate the current menu. |
| F6/18 | post message mail waiting or not waiting. |
| F12/24 | displays the previous command in the command area Up to 5 commands can be recalled. |
| PA1 | terminate the current menu. |
| ENTER | process the command area. |

# Chapter 11. Editor and REXX Considerations

## Editor Considerations

The file used for executing the Editor is $PGM:EDITOR. It must have the public and execute-only attributes since the Editor requires certain privileges. You may wish to change this file to redefine the default region size, work file size, function key definitions, and macro library, or issue commands during the edit startup. Bear in mind that this file contains the system-wide defaults. If an individual user has special requirements, they can be met by creating a personal Editor file for that user, rather than changing the defaults for everyone.

There is a similar file, $PGM:BROWSE, used by the BROWSE command.

### Region Size

The default region size of 400K is adequate for editing all files. Certain editor macros may require additional memory.

### Work File

The EDITOR requires a work file approximately twice as large as the file being edited. The standard size of 6000 records of 128 bytes each allows you to edit a file about 4500 80-byte records. Generally this is sufficient for most users and, when it is not, a personal Editor file with a larger work file solves the problem. The work file space is allocated from the system scratch dataset (SYS1.MUSIC.SCRATCH). If space is limited in this dataset, you may wish to define a smaller work dataset for the default Editor. The BIGEDIT command (file $PGM:BIGEDIT) uses a larger work file.

### Commands and Function Keys:

Editor commands can be placed after the /LOAD EDIT statement. These will be issued each time the Editor is started. This is a useful place to define any local function key defaults that are different from those distributed with the system. In order to be consistent with the standard Editor, a "TOP" command should be issued last, to position the line pointer and place the cursor in the command area.

### Editor Configuration Switches

There are some option bits at displacement 000C in Load Library member EDITOR. They can be REPed in order to customize some characteristics of the editor. In the distributed editor, all the bits are off. For compatibility with other MUSIC sites, it is recommended that you not change these bits. However, the choice is yours. The bits are defined near the beginning of module EDITIO ($EDT:EDITIO.S).

The bits are as follows:

Bit x'80' If this bit is on, the editor will start with an empty file (as if the NEW option was specified), if the specified file does not exist. If this bit is off, the editor displays an error message and exits if the file does not exist and NEW was not used.

Bit x'40' If this bit is on, the automatic TEXT UC done by the editor in some cases when it starts, is

suppressed.

Bit x'20' If this bit is on, the automatic TEXT LC done by the editor in some cases when it starts, is suppressed.

Other bits: Reserved.

Use the SYSREP utility to change the bits, by running the following job:

```
/INC SYSREP
LIB='SYST'
NAME EDITOR
VER 000C xx
REP 000C yy
```

where xx is the current hex value of the byte and yy is the new hex value of the byte. The first time you do this, xx is 00. After the first time, xx is the current hex value of the byte (the previous yy). yy is the sum of the bit values you want to be on. For example, to set bit x'80' only, yy is 80. To set bits x'80' and x'20', yy is A0. To turn off all the bits, yy is 00.

*Notes:*

1. The change will take effect at the next IPL of MUSIC.

2. These option bits apply to all edits for all users.

3. They are valid for MUSIC/SP 2.2 and all later releases.

4. You need to reapply the REP after any service fixes that replace the EDITOR member in the Load Library, and after upgrading or reinstalling MUSIC.

5. The displacement 000C will not change.

## Editor Macros and Multi Tasking

Editor macros can be written in REXX. When a command is entered that is not a standard editor command it is passed on to the REXX interpreter. The interpreter scans MACLIB PDS for the macro. If found, the macro is read into memory. Editor commands in the macro are passed back to the editor for execution. These may in turn invoke other macros recursively.

If a macro is not found in MACLIB, the command is executed as a MUSIC/SP command concurrently with the edit session, using the multi-tasking facility. Using this, a number of files can be edited at the same time and the multi-session function keys used to switch between edit sessions. The following commands effect the editor's processing of macros and commands.

REXX on/off

Used to enable or disable macro processing.

CMDS on/off/none/macro

Used to control whether multi tasking execution is allowed. There are a number of sample macros on the userid $EDT. They are not documented and are only intended to serve as samples for users who wish to write their own macros. Complete documentation on writing Editor macros is included in the *MUSIC/SP User's Reference Guide*.

| | |
|---|---|
| KEYS | - define program function keys |
| BROW | - browse a file while editing |
| FE | - edit certain fields in a file |
| GET | - get a new file for editing |

## REXX Considerations

The file $REX:REXX contains the default control statements for running the REXX interface. This file must be public. You should specify a region size that is suitable for the execution of any REXX programs that are available for general use at your site. Mixing REXX programs with different region sizes specified, can cause errors. To avoid problems, those writing REXX programs should not specify a region size in the programs themselves, but should use a /INCLUDE REXX statement at the beginning of the programs to pick up the system default region size.

The default is set at 1024K which is sufficient to run the Full Screen Interface (FSI) and other REXX applications that come with the system. If an individual requires more memory, create a private file called REXX in the user's library that specifies a larger region. Note that well behaved programs that start with "/INCLUDE REXX" will, in this case, pick up the region size from the user's private REXX file.

When a MUSIC/SP command is issued from a REXX program, the program is terminated, the command executed and the program restarted from where it left off. Because the restart is done from a work file and not the original file, any program privileges assigned to the original file are lost. Components of REXX programs that require privileges should be written as separate commands and invoked as required from the main program.

# Chapter 12. TODO Facilities

## REMIND Facility

The REMIND program allows you to set reminders of a future event. Reminders appear on the screen when you request the TODO facility.

### Program Parameters and Authorization Tables

The executor file for REMIND, $TDO:REMIND, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

**Example:**

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CANCEL=t or f When CANCEL=t is specified the user can return to *Go by PA1, /CANCEL ALL, or successive END commands or PF3's. However when CANCEL=f, /CANCEL is rejected and normal termination of REMIND results in a sign-off. In other words CANCEL=f prevents users from returning to *Go. The default is CANCEL=t.

ERRFIL='filename' Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:REMIND.MSGS. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:REMIND.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:REMIND.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

FILTER=t or f When FILTER=f, any command string which is not an option code or a command used by REMIND is passed to the MUSIC command processor. When FILTER=t,

the option codes, the commands used by REMIND, and only the commands and/or programs listed directly below the parameter line are allowed. The default is FILTER=f.

A number of commands processed by the MUSIC terminal scanner can not be issued from REMIND. These commands are: /COMPRESS, /CTL, /DISCON, /NS, /PAUSE, /PROMPT, /REQUEST, /RUN, /STATUS, /TIME, /TEXT, /TABIN, /TABOUT, /USERS, /VIP, and /WINDOW.

The commands can be entered with any abbreviation up to the minimum abbreviation as specified by an integer to its right. If no minimum abbreviation is specified the command will have to be entered in its entirety as specified. Before the command string is passed the abbreviated name is expanded to its full length. For example, the following abbreviations would be allowed for this list of commands:

| Command | Abbreviations |
|---|---|
| PURGE 2 | purg, pur, pu |
| OUTPUT 3 | outpu, outp, out |
| SUBMIT 2 | submi, subm, sub, su |
| PRINT 3 | prin, pri |

If no commands and/or program file names are specified, then the "/" command is not allowed at all. This is the method for barring the use of the "/" command. For example, in the file ABC:

```
/INC REMIND
MYNAME='ABC',FILTER=T
```

ABC will not allow the use of the "/" command. The stack file will be @REMIND.STAK.tcb, where *tcb* is the TCB number.

FS=t or f

When FS=t and REMIND is invoked on a 3270-type terminal, automatic full screen support is provided. On a non full screen terminal such support is never allowed.

When FS=f, no matter what terminal type you are using, only non full screen support will be available. The default is FS=t.

KEYFIL='filename'

Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

The user can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:REMIND.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:REMIND.KEYS have the same format as the lines in KEYFIL filename.

LOG=t or f

The LOG=t parameter can be used if you wish to keep a usage log. LOG=f is the default.

LOGFIL='filename'

Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@REMINDLOG.tcb, where *tcb* is the current TCB number.

MDELAY=n — Where *n* is a value indicating the delay between attempts at finding out if mail is waiting. When mail is detected, a mail waiting message is posted. When *n* is a positive integer it is taken as representing minutes. When *n* is negative it is taken as seconds. To stop all mail checking, use a value for *n* that is greater than 24 hours such as -86400 secs or 1440 mins. This will prevent the posting of even the initial mail waiting message at the start of the program. The default is MDELAY=-300 or MDELAY=5 (five minutes).

MYNAME='filename' — Where *filename* is the name of the file. In our example it would be ABC. If this parameter does not specify the file name in which it is stored REMIND will not make the file an *always* program.

STACK='filename' — Where *filename* is a 1-46 character name of a file to be used as the stack file. The tcb number is appended to it as follows: *filename.tcb*. The stack file is used to store a trace of menus and programs that have been called. In this way the program can return to the correct menu as it drops back through the stack. When no stack file is specified the default "@REMIND.STAK.tcb" is used, where *tcb* is the TCB number.

*Note:* Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

# Schedule and Meet Facilities

The SCHEDULE program allows you to organize your agenda on a daily or monthly basis. It can be used to schedule conference rooms or equipment items also.

The MEET program allows you to schedule a meeting for a group of attendees, conference rooms, and equipment items.

The CONFLIST program allows you to create a conference room or equipment item and gives authorization to a number of users. (See the *MUSIC/SP Office Applications Guide* for a user description of this program.)

All three programs share the same schedules (files) when they are used.

The SCHEDULE and MEET programs are configured with the MAIL.CONFIG program. See the MAIL.CONFIG program description in this manual for further details.

## Program Parameters and Authorization Tables

### SCHEDULE Parameters

The executor file for SCHEDULE, $TDO:SCHED, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

CONFCD='userid' — A file holds the conference rooms and equipment items information as given by the CONFLIST program. CONFCD tells the program the userid where this file is stored. The default for CONFCD is $TDO.

ALIAS='name'
MUSNOD='name' — These are the names of the local MUSIC system; example MUSNOD='MUSIC'. See MAIL parameters under the topic "Configuring MAIL" in *Chapter 9* -

*Electronic Mail Facility* for a detailed description of this parameter.

**MEET Parameters**

The executor file for MEET, $TDO:MEET, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

The program parameters are as follows:

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests this facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If ACCESS='filename' is not specified, then all userids have access to this facility.

**Example:**

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CONFCD='cod1','cod2','cod3','cod4','cod5'
Where *cod1, cod2, cod3, cod4, cod5* are valid MUSIC userids. These userids are used to access their respective conference rooms/equipment items files, better known as @CONFEQUIP. This can be used to have different department's conference rooms/equipment items accessible for meeting scheduling from one program. The default CONFCD userid is $TDO.

DEFTM1=n This parameter specifies the default BEGIN TIME for the program MEET. DEFTM1=900 (9:00 am) is currently specified for this program. This default BEGIN TIME is used when the user intends to schedule a meeting for a day other than the current day. If the current time is prior to 700 for the current day, then DEFTM1=900 is used. If it is later than 700 for the current day, then the MEET program inserts a BEGIN TIME equivalent to the next hour plus one hour. The value of *n* must be greater than or equal to 1 and less than or equal to 2400.

Example 1

If the current time is 1015 when a user is scheduling a meeting, then the BEGIN TIME will be 1200 on the screen.

Example 2

If the current time is 610 (6:10 am) when a user is scheduling a meeting, then the BEGIN TIME will appear as 900 (DEFTM1=900).

DEFTM2=n

This parameter specifies the default END TIME. This time appears on the screen unless the user specifies an END TIME. DEFTM2=1700 is currently specified for this program. The default END TIME is also used for multiple day meetings as the END TIME for all days of the meeting except the last day. The value of *n* must be greater than or equal to 1 and less than or equal to 2400. The value of DEFTM1 must not be greater than the value of DEFTM2.

Example 1

If the current time is 1401, then the default END TIME will appear as 1700 on the screen. (The BEGIN TIME will be 1600.)

Example 2

If the current time is 1800 on April 2nd, then the date changes to the next day and the values for both DEFTM1 and DEFTM2 appear on the screen:

```
Meeting Information
  Begin date    ===> 03APR86 (ddmmmyy)
  End date      ===> 03APR86 (ddmmmyy)
  Begin time    ===> 900____ (hhmm)
  End time      ===> 1700___ (hhmm)
  Time required ===> __ (hours) __ (minutes)
```

ERRFIL='filename'

Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:MEET.MSGS. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:MEET.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:MEET.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

KEYFIL='filename'

Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

Users can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:MEET.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:MEET.KEYS have the same format as the lines in KEYFIL filename.

LOG=t or f

The LOG=t parameter can be used if you wish to keep a usage log for the program MEET. LOG=f is the default.

LOGFIL='filename'

Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@MEETLOG.tcb, where *tcb* is the current TCB number.

ALIAS='name'
MUSNOD='name'

These are the names of the local MUSIC system; example MUSNOD='MUSIC'. See MAIL parameters under the topic "Configuring MAIL" in *Chapter 9 -*

*Electronic Mail Facility* for a detailed description of this parameter.

STACK='stackname' A 1-46 character name of a file to be used as the stack file. The TCB number is appended to it as follows: *stackname.tcb*. MEET establishes an always program environment. The stack file is used to restore the environment when the *always* program MEET is re-entered. The default stack name "@MEET.STACK.tcb" is used when no stack name is specified.

*Note:* Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

**CONFLIST Parameters**

The executor file for CONFLIST, $TDO:CONFLIST, contains some program parameters (in MUSIC name-list format, separated by commas) which you may need to change.

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

**Example:**

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

ACODE='cod1' Where *cod1* is the MUSIC userid used to find the conference rooms and equipment items file, better known as @CONFEQUIP.

ERRFIL='filename' Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:CONFEQUIP.MSGS The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:CONFEQUIP.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:CONFEQUIP.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

### Special File Names

The following special files are created by SCHEDULE, MEET, and CONFLIST at various times.

@MMMYY.SCHED. actual user's calendar of events for month MMM given the last two digits of the year YY. A separate file exists for each month. The last 12 months are kept on the system.

@MMMYY.NAME monthly schedules for conference room or equipment item NAME, MMM and YY being the month and last two digits of the year. A separate file exists for each month. The last 12 months are kept on the system.

@AUTHSCHED list of users authorized to look at, add, or update the schedule.

$TDO:@SCHEDMEET.TABLE
file used to control the access to schedules when using the MEET program. Enqueuing and dequeuing are done with this file.

@CONFEQUIP list of conference rooms and equipment items. 125 rooms and items can be accommodated. This file contains the conference room or equipment item name, an administrator userid, a description of the room or item, a list of users authorized to access this item.

@TMENULOG.tcb log file used in TMENU

@REMINDLOG.tcb logfile used in REMIND

@MEETLOG.tcb log file used in MEET

@TMENU.STACK.tcb
stackfile used in TMENU

@REMIND.STAK.tcb
stackfile used in REMIND

@MEET.STACK.tcb stackfile used in MEET

---

## Installing a System Word Dictionary in the PLPA

A system word dictionary can either reside in the PLPA (pageable link pack area) or in the link pack area itself. This topic describes how to prepare a dictionary and install it in the PLPA.

When a dictionary is in the PLPA it is available to any user who has access to the system. However it does not occupy the user region. In this way many users will share the dictionary while each will run in a substantially smaller region than otherwise would be required.

MUSIC is distributed with a system word dictionary in the PLPA called DICT1. References will be made to this dictionary in order to illustrate how the installation is done. You should note that this procedure need only be followed if you want to add a new system dictionary or modify DICT1.

**This procedure must be performed by system support personnel.**

## Initial Preparations

The SPELL algorithm requires that all words in the system dictionary be in lower case. Only letters of the alphabet (a-z) are permitted. Any special characters are translated out. Therefore when preparing a new list of words or modifying an old one, please remember to remove special characters. For example if you want to have *didn't* in your system dictionary you need to have the two entries "didn" and the letter "t". Spell ignores the "'" (quote) and sees two words "didn" and "t". In this way it will not flag *didn't* as an error. In addition numeric text is also ignored. For example if "pf" were an entry in the system dictionary then "pf1", "PF12", or "pf" would all be correct spellings.

You should also make sure that each letter of the alphabet is present in the dictionary, since the occurrence of any single letter is a correct spelling. This also ensures that abbreviations of the type A.B.C. are spelled correctly since the periods are translated out.

Several steps have to be performed to install a system dictionary in the PLPA. These are:

1. compute the PLPA storage required
2. process and compress the dictionary
3. compile the dictionary
4. link edit the dictionary
5. place the dictionary's name in the system catalog
6. install the dictionary in the PLPA

Descriptions of these procedures follow below.

*Note:* The raw word files for the system dictionary DICT1 are

$TDO:WORDS.A,
$TDO:WORDS.B,
$TDO:WORDS.C,
... to
$TDO:WORDS.Z,
and
$TDO:WORDS.APROPER,
$TDO:WORDS.BPROPER,
$TDO:WORDS.CPROPER,
... to
$TDO:WORDS.ZPROPER.

## 1. Compute the PLPA storage required

After you have prepared the list(s) of words for your dictionary, it is necessary to compute the number of total characters or bytes it will occupy in the PLPA. This is done by processing the list(s) through the BYTES program. This program will report the total number of lines it read (words in the dictionary), the average length of each word, and the total bytes the words would occupy if totally compressed and contiguous.

It is important that you pay attention to this step and the results it produces. You may have to modify source programs in later steps to accommodate very large dictionaries. The programs to be used later, PROCESS and COMPDICT, are setup to handle a dictionary of 830,000 bytes.

The BYTES program can be invoked as follows:

```
/INC BYTES
/INC words
```

where *words* is a file name containing the dictionary. Any number of files can be included here. A file called $TDO:BYTES.RUN1 has the run used for DICT1.

## 2. Process and Compress the Dictionary

After you have verified that the total number of characters in the dictionary is less than or equal to 830,000 you can proceed to running the PROCESS program. If however you have more than 830,000 bytes, you must modify $TDO:PROCESS.S and $TDO:COMPDICT.S.

In $TDO:PROCESS.S change the statements:

```
REAL*8 DICT8(102500)
LOGICAL*1 DICT1(820000)
```

to fit your needs. DICT1 must be a multiple of 8 and DICT8 must be dimensioned as the dimension of DICT1 divided by 8. Failure to ensure this will cause compiler errors.

In COMPDICT change the statements:

```
LOGICAL*1 ALL(965000)
LOGICAL*1 DICT(960000)
```

to fit your needs. The 5000 discrepancy between ALL and DICT must be maintained in any change you make.

In addition you should make sure that the /SYS REG=1000 in $TDO:PROCESS and $TDO:COMPDICT is incremented accordingly.

If you made changes to PROCESS.S or COMPDICT.S you must execute and relink them. Files $TDO:PROCESS.LKED and $TDO:COMPDICT.LKED should be used to generate a new load module. You can now continue with the processing of the dictionary.

PROCESS performs several functions and provides information required by SPELL. The first function is to translate out any non-alphabetic character in a word, the word is right justified and converted to lower case. Then each word is analyzed to determine the frequencies of character pairs as they occur and stored in $TDO:PAIRS.DICT. This information is used later by SPELL to aid in finding alternate spellings when words are not found in the dictionary. The total list is then sorted. The sorted list is converted to a compressed and contiguous list of characters and stored in $TDO:COMPRESS.DICT. In $TDO:MARKERS.DICT is stored the byte addresses for the dictionary, later to be used for accessing the list. The file $TDO:PROCESS.RUN1 has the run for DICT1.

PROCESS requires a region of 1000K bytes, and because of the extensive processing of each word, it will require a great deal of processing unit time. This process time is in the order of 1200 su for 830,000 bytes.

The following is a list of $TDO:PROCESS.

```
/SYS REG=1000,TIME=50
/FILE LMOD N($TDO:PROCESS.LMOD) SHR
/FILE 1 UDS(----TEMP) NREC(125000) LR(32) VOL(MUSIC3) NEW DELETE
/FILE 2 RDR
/FILE 3 UDS(----TMP3) NREC(80000)  VOL(MUSIC3) NEW DELETE
/FILE 8 N($TDO:MARKERS.DICT) NEW(REPL) DEF
/FILE 10 N($TDO:COMPRESS.DICT) NEW(REPL) SP(300) RLSE DEF
/FILE 11 N($TDO:PAIRS.DICT) NEW(REPL) SP(100) DEF
/LOAD EXEC
```

```
PROCESS
```

## 3. Compile the Dictionary

After the dictionary has been successfully processed by the PROCESS program, it can be compiled into an object deck. COMPDICT will take the $TDO:PAIRS.DICT, $TDO:MARKERS.DICT, and $TDO:COMPRESS.DICT and generate an object deck. The file $TDO:COMPDICT.RUN1 was used to generate DICT1.

The following is a listing of $TDO:COMPDICT.

```
/SYS REG=1000,TIME=MAX
/FILE LMOD N($TDO:COMPDICT.LMOD) SHR
/FILE 1 N($TDO:COMPRESS.DICT) OLD DEF
/FILE 2 N($TDO:MARKERS.DICT) OLD DEF
/FILE 3 N($TDO:PAIRS.DICT) OLD DEF
/FILE 7 N($TDO:DICT.OBJ) NEW(REPL) SP(500) DEF
/LOAD EXEC
COMPDICT
```

## 4. Link Edit the Dictionary

In order for the dictionary to be placed into the PLPA it must be link edited into a load module. The file $TDO:DICT1.LKED was used to generate DICT1.

The following is a listing of $TDO:DICT1.LKED

```
/SYS TIME=MAX,REG=200
/FILE LKEDWORK NR(16000)
/FILE LMOD NAME($TDO:DICT1.LMOD) NEW(REPL) SP(1000)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
/INC $TDO:DICT1.OBJ
 NAME DICT1
```

This step produces the file $TDO:DICT1.LMOD. This is the file that is to be loaded into the PLPA. Note that the object deck it requires is $TDO:DICT1.OBJ that was produced by $TDO:COMPDICT.RUN1.

## 5. Place the Dictionary's Name in the System Catalog

To install a dictionary in the PLPA, edit the system catalog and add a line like the following:

```
----+----1----+----2----+----3
RESPGM64 DICT1    PLPA 300000
```

Make sure that the number for the RESPGM keyword is unique. If RESPGM64 is already being used by some other module, change the 64 to a number not currently being used.

Once you have modified the catalog, you must run the EDTCAT utility as follows:

```
/INC *COM:EDTCAT
/INC xxxxxx
```

Where **xxxxxx** is the name of your system catalog. (Most installations use the name "$GEN:SYSCAT" for the system catalog. You need the VIP privilege on your userid to execute this utility. When you next IPL MUSIC, the dictionary will be accessible from the PLPA.

## 6. Install the Dictionary in the PLPA

The $TDO:DICT1.LMOD will be placed into the PLPA using the file $TDO:DICT1.PUTSYS. The following is a listing of $TDO:DICT1.PUTSYS

```
/FILE 1 N($TDO:DICT1.LMOD)
/INC LDLIBE
LIBE='SYST',IN=1
NAME='DICT1',RENT=T,REPL=T
```

As a final step the name DICT1 has to be entered in the system catalog and becomes available at the next IPL. If it is not listed here it will remain inaccessible even though it has been properly installed.

## Accessing the Dictionary

In order to access a specific system dictionary you must set up a file as follows:

```
/INC SPELL.PROG
name
```

Where *name* is a 1-8 character name corresponding to a system dictionary in the PLPA. If name is not specified it defaults to DICT1. If the file from above were called SPELL2 then the dictionary specified by *name* is accessed by typing SPELL2 when in *Go mode. See the contents of files $TDO:SPELL.PROG and $TDO:SPELL as examples.

# Chapter 13. IIAS/IIPS

## IIPS/IIAS Codes

IIPS/IIAS uses special codes to identify authors, students and administrators. These codes have no connection with the MUSIC sign-on userids. The student codes start with the letter S, the author codes with the letter A and the administrator has the code of XUPER.

When the author signs on to a course, the course can be altered or author-as-student mode can be entered to check out the course. The administrator code is used to allocate course areas, register students and perform dump and restore functions over any of the course files.

## Course File Organization

A single course disk file can contain many individual courses. Each course is allocated a specified number of blocks within this disk file. Within each course area is space to hold information about each student that is registered for that course. The IIPS student code is used to point to the user's specific status area within the course space. These status records allow continuation of the course through many terminal sessions.

The course author must have write access to the course disk file. The student needs write access to the student's own student status area. In most IIPS implementations, this means that the student also needs write access to the course disk. Control of the student write access to file can be performed in one of two ways on MUSIC: *Multiple Write Access* and *No Write Access*.

### Multiple Write Access

This method allows the student to write directly to the student's own status area within the course disk file. Each student must be individually authorized so that a specific area can be reserved for the student within the course. The MUSIC disk file that contains the course must be defined so that the student users can have write access to it. The OPTIONS file (described later) should be used to prevent two users from attempting to use the same IIPS sign-on code simultaneously or from attempting to sign on as the author or administrator.

### No Write Access

The MUSIC implementation of IIPS allows the student status area to be located on a separate file. This allows the course file to be defined with only read access from students. This provides additional protection against accidental or deliberate alteration of the course material by unauthorized users. Furthermore, it allows any user to take the course without being individually preregistered. In this case, all students must sign on to the course with the special code of STUDENT.

# File Usage

A number of disk files can be defined for use with IIPS/IIAS. Most are created and initialized by the utility program IIS.INIT. A file with the necessary control statements is set up to point to these files. This file should be saved with the execute-only option. After this is done, simply type the name of the file to invoke IIPS/IIAS with your files.

It is wise to set up a file for student users and a separate one for course authors and administrators. This separation can provide for better control of write access to the files.

The file will be of the form:

```
/SYS NOPRINT,TIME=MAX
/FILE ..... (as required to point to the IIS files)
      ..... additional /FILE statements as required
/INCLUDE IIS
```

The /FILE statements are described in the following topics. Reference to CODE should be taken as referring to the first 4 characters of the MUSIC sign-on code of the file's owner.

## Special note on UDS files

When UDS files or tapes are used with IIPS/IIAS the parameter BUFNO(2) must be specified on the /FILE statement. Failure to do so will result in errors during course execution.

## Course Files

Course files are defined via /FILE statements that have ddnames of DISK20 through DISK59. (DISK60 through DISK69 are reserved for possible system use. Currently DISK69 is used for the IBM supplied course and macro library and DISK68 is used for the MUSIC LEARN courses.) First estimate the number of blocks needed by consulting the writeup under the REGISTER COURSE command in the *IBM IIPS/IIAS Administrator's Guide* (SH20-2449).

IIPS/IIAS should always access a course file through the same ddname. In other words do not define a course file as DISK30 one time and DISK31 a later time.

Course files up to 3900 blocks can be stored in the Save Library. To create and initialize one simply type IIS.INIT when your terminal is in *Go mode and respond to the questions it will ask.

Large course files can be set up in UDS files. The maximum number of blocks per disk file is limited by IIPS/IIAS to be 32700 blocks. To allocate and initialize the UDS file, execute a file containing the following:

```
/FILE 1 UDS(dsname) LRECL(512) NREC(n) VOL(MUSICx) NEW
/INCLUDE IIS.INIT
```

In the above, *n* is the number of course records. The 5th character of the dsname field must be a "#" character if the file is to be writable from any MUSIC user code.

Sample /FILE statements for course files on files are:

```
/FILE DISK30 NAME(MYCOURSE) SHR   (no write access)
/FILE DISK30 NAME(MYCOURSE) WSHR  (multiple write access)
```

Sample /FILE statements for course files on UDS files are:

```
/FILE DISK30 UDS(CODEII30) SHR VOL(MUSIC1) BUFNO(2) (no write access)
/FILE DISK30 UDS(CODE#I30) WSHR VOL(MUSIC1) BUFNO(2) (write access)
```

## Student Recording Files

Records can be kept of student responses. (The utility programs used to analyse this data are described later.) This optional file is initialized via the IIS.INIT program.

A sample /FILE statement for a message file CODE:MYSREC.IISR follows. Note that the letters IISR are to be replaced by * on the /FILE statement. The /FILE statement for TEMPSREC must also be given as shown though it never needs initialization.

```
/FILE MAINSREC PDS(CODE:MYSREC.*)
/FILE TEMPSREC NAME(&&TEMP) SPACE(2) MAXSP(2) RECFM(VC) NEW
```

## Student File

This file is used when the student status information is not to be written to the course file. (Do not confuse this file with the student recording file previously described.) A student code of STUDENT must be registered for the course and the student must sign on with STUDENT before this file will be used. The name of the file can be given as &&TEMP if the student is to start at the beginning of the course each time it is used.

Use the name @WARM (for example), to keep the student's status for the student's next use of the course. Thus, the point where the student left off in the course can be continued. If the student tries a different course next time, the warm start information for the first course is automatically erased. Each student should have a different MUSIC 4-character code so that each can get a unique warm start area.

The warm start information is dependent on the disk location of the student records. Thus if the course is moved from one disk to another, the warm start information will not be used causing the student to start at the beginning of the course.

A sample /FILE statement for a student file called @WARM is:

```
/FILE STUDSAVE NAME(*USR:@WARM) RECFM(U) SPACE(4) OLD(CREATE)
```

## Options File

This optional file contains special run options. Create the options file by using the Editor and save it public. Each option is entered on a different line in the file.

NXTPGM=code:name  This option specifies that file code:xxxx will be run after IIPS/IIAS stops for any reason. The program will be invoked with the non-cancelable option. The given name may even be the file that invokes IIPS/IIAS. If this file is also the name of the MUSIC code's AUTOPROG then the user will always be in the IIPS/IIAS environment when the user uses this code. The user may type /OFF at IIP/IIAS sign-on time to disconnect from MUSIC.

ENQPRE=xx — This option will perform a system enqueue operation using the IIPS/IIAS sign-on code prefixed with the two letters xx. If this name is already used, the sign-on will be rejected with the message `STD# IN USE..TRY AGAIN LATER.` This ENQPRE option has no effect when the code STUDENT is used and a /FILE statement for STUDSAVE is present.

A=code — This option checks to ensure that authors are signed on with the specified MUSIC code. Its use is mainly to protect files that are public writable. Several A=code statements can be used if more than one MUSIC code is to be allowed. If no A=code statements are used no MUSIC code checking is done. The length of the code given on these statements is the actual length compared. The message `UNAUTHORIZED TO USE THIS IIS CODE` will be issued if the code check fails.

X=code — Same as A=code except it gives the valid MUSIC codes for IIPS/IIAS administrators. The IIPS/IIAS code of XUPER is the only accepted administrator code.

A sample /FILE statement for an options file CODE:MYOPTS is:

```
/FILE OPTIONS NAME(CODE:MYOPTS) SHR
```

### Location Table

This is used to store communications between the various users of the instructional system. These communications include course comments from students as well as messages from authors or administrators using the LOG command. This file also contains the location table used to verify student location during registration. By default, MUSIC provides a dummy file, resulting in a warning message during the *register student* process, and the loss of messages and comments. The IIS.INIT program can be used to create a location table file if one is required.

Sample /FILE statement for the location table file MYLOC is:

```
/FILE LOCTAB NAME(MYLOC) WSHR
```

## Preparing the Course Material

Refer to the IBM publication *IIAS Course Authoring Guide* (SH20-2450) for the details of how to prepare and modify courses.

Course materials may be prepared outside of the IIPS/IIAS environment. This is done by using TEDIT to invoke the MUSIC's Editor with the option to preserve lower case letters. Do not enter TAB characters in course materials. Use files with names of 8 characters or less. Input these files into IIPS/IIAS by using the GETX command.

Use the sequence of two at signs (@@) to signify the entry of a carriage return when entering Coursewriter text.

The use of the Course Structuring Facility (CSF) is recommended to greatly ease the production of courses. A good overview of CSF is presented in the IBM publication *IIPS/IIAS General Information Manual* (GH20-2446). The user should note that the powerful CSF facility does take a fair amount of computer time to generate complete courses. Minor modifications of generated material presents no hardship. Execution speed of these generated courses is not slower by any noticeable amount.

## Author Commands

The following are some notes about some IIPS/IIAS author commands. IIPS/IIAS limits external file names to 8 characters or less. These commands can be entered only in response to the TYPE COMMAND message.

| | |
|---|---|
| getx aaaaaaaa | This command is used to read the file *aaaaaaaa* into IIPS/IIAS just as if you have typed its contents live at this point. When the end of the file is read, you may continue entering commands. If the file is not accessible, no action is performed and no message is given. |
| get aaaaaaaa | This command is the same as getx with the exception that the terminal output will be sent to the file IISOUT instead of to the terminal. |
| put bbbbbbbb | This command directs that the terminal output will be sent to the file *bbbbbbbb* instead of to the terminal. |
| putx bbbbbbbb | This command directs that the terminal output associated with TYPE commands will be sent to the file *bbbbbbbb*. All other terminal output will be ignored. The problem and sequence numbers will be stripped from the output. This allows the author to create a file that can be used with the Editor to make material changes. Reinsertion can then be done by the get or getx commands. |
| poff | This command stops the output to disk operation started by a put or putx command. |
| incore<br>store<br>nostore | These commands are not used because MUSIC automatically buffers disk I/O. |
| comments<br>log | These commands are not used in the MUSIC environments. |
| regs | As this file requires the HISTORY file, it will not work on MUSIC. |

## Administrator Commands

The following are some notes about some administrator commands. The IBM publication *IIPS/IIAS Administrator's Guide* (SH20-2449) documents these commands. No secondary authority is required for any command. The get, getx, put and putx work in administrator mode as they do in author mode.

Some commands refer to work files. These normally refer to files of RECFM V or VC. Thus work11 is the file WORK11. If the file does not exist, one will be created. The file may refer to a tape file during a course on or off process. In this case, run the job from batch and use RECFM=U on the TAPE /FILE statement.

A number of commands have meaning only when HISTORY files are used or when IIPS/IIAS is directly managing the terminal network. The meaningful commands in the MUSIC environment are:

COPY
COURSE OFF
COURSE ON
COURSES

DELETE COURSE
LIST
LOC (useful only when a LOCTAB file used)
MODIFY STUDENT
PRINT
REGISTER COURSE
REGISTER STUDENT
REMOVE COURSE
REMOVE STUDENT
REORG
SEE DISK
SEE STUDENT
STUDENT STATUS
TRACE
TYPE
UPD
WRITE DISK
WRITE STUDENT

# Notes

IIPS/IIAS automatically puts a read up on the terminal if an attempt is made to print more than about 100 lines without an intervening read. The entry of a blank line at this time will cause the output to continue. The entry of QUIT in author mode will stop the command. The entry of SIGN OFF will also be accepted at this time.

IIPS/IIAS accepts a line delete signal of 3 number signs (###) when entered at the end of a line.

# Utilities

A number of utility programs are provided with IIPS/IIAS. Some examples of their output are given in the two IBM publications: *IIPS/IIAS Operations Guide* (SH20-2459) and *IIPS/IIAS Course Authoring Guide* (SH20-2450).

## Analyzing the Student Records

The student record analysis utilities produce reports on student responses and sign-on/sign-off activity. The analysis is performed by running a set of jobs. Normally these jobs are run at batch due to the amount of output generated.

Input to these jobs is the student recording file. The recording option specified on the REGISTER STUDENT administrator command affects what records will be written to this file.

If the special code of STUDENT was used, the student code reported in the reports will be of the form "S*code" where *code* is the user's MUSIC sign-on code.

```
Job 1:      /FILE FILE10 NAME(CODE:MYSREC.IISR) SHR
            /INCLUDE IIS.SRAJ1
```

Job 2: `/INCLUDE IIS.SRAJ2`

Job 3:
```
/INCLUDE IIS.SRAJ3
...control statement
```

The format of the control statement is:

| | |
|---|---|
| Cols 1-8 | defines course selection information. |
| Col 1 | controls the translation to lower case. If 1, then a translation will be made to lower case. |
| Cols 2-7 | contains the course name. If blank all courses are selected. |
| Col 8 | used when a course name is specified, a 1 causes all courses whose names precede the selected course in sorted sequence to be selected. |
| Cols 9-19 | defines student selection information. |
| Cols 9-18 | contain the student number. If blank all students will be selected. The student number must be prefixed with the letter s. |
| Col 19 | used when a student number is specified, a 1 causes all students whose student number precede the selected student number in sorted sequence to be selected. A blank will cause only the specified student to be selected. |
| Cols 20-25 | selects the type of response to report. |
| Col 20 | set to 1 to select UN response records |
| Col 21 | set to 1 to select HELP response records |
| Col 22 | set to 1 to select GO TO response records |
| Col 23 | set to 1 to select CA and CB response records |
| Col 24 | set to 1 to select WA and WB response records |
| Col 25 | set to 1 to select AA and AB response records |
| Cols 27-30 | defines which course parameters to include in the report |
| Col 27 | set to 1 to cause parameters and switches to print |
| Col 28 | set to 1 to cause counters 1-10 to print |
| Col 29 | set to 1 to cause counters 11-20 to print |
| Col 30 | set to 1 to cause counters 21-30 to print |

Job 4: `/INCLUDE IIS.SRAJ4`

Job 5: `/INCLUDE IIS.SRAJ5`

Job 6:
```
/INCLUDE IIS.SRAJ6
...location statement(s)
...date statement
```

Up to 24 location cards may be given. The format of each is:

| | |
|---|---|
| Cols 1-2 | area number |
| Col 3-19 | location name (any printable characters will do) |

The format of the date card is:

| | |
|---|---|
| Col 1 | must be a * to identify it as a date card |
| Col 2-26 | date heading (e.g., JUNE 23, 1978) |

## Getting Course Batch Listings

To obtain a course listing at batch first use the administrator PRINT command documented in the *IIPS/IIAS Administrator's Guide* (SH20-2449).

For example, the administrator can issue the following commands:

```
print 11    (to output the file to the file WORK11)
cname
sign off
```

Once this has been done run the following jobs from batch:

```
Job 1:      /FILE TINN  NAME(WORK11) SHR
            /INCLUDE IIS.CBLJ1

Job 2:      /INCLUDE IIS.CBLJ2
```

## Converting ITS and Coursewriter Course Material

The course conversion program converts ITS and Coursewriter course materials into a format acceptable to IIPS/IIAS. The utility accepts one course file per tape. If the tape is blocked, it should be first copied to a file.

```
/PARM cname 00
/FILE PRINTIN TAPE ....
/INCLUDE IIS.CCONV
```

The /PARM statement contains the 6 character course name in lower case letters followed by the segment number (normally 00). If the course name is less than 6 characters, put in blanks after such that the segment number always starts 6 characters after the course name.

After the course has been converted, register the course, sign on as the course author and fetch the file by using the get or getx author commands. Specify a file name of *convcrse*. (Purge this file after you have finished with it.)

# Installing New Courses From Tape

The main problem in installing new courses is that they can be stored on tape in such a variety of ways. Also, some courses require that new subroutines or screen formats be installed along with the course material. These may be in source or object deck form. The key to the problem is selecting a method for getting the material off the distribution tape. The following MUSIC utilities can be useful to this end.

| | |
|---|---|
| CMSTAPE | - cms tape dump format |
| GENSAV | - iebupdte format |
| LOADPDS | - iebcopy format |
| MOVEPDS | - iehmove format |
| TAPUTIL | - straight block by block copy |
| DBLOCK | - deblock tape file to 512 byte record on disk |
| UTIL | - record by record copy |

The three topics which follow illustrate the steps involved in installing an actual course, installing some new functions, and installing new screen formats.

# Installing New Course Material

Courses obtained externally (from IBM or other users) generally come on magnetic tape in one of three forms:

Course off
Print
Insert

The course tape package should include the following information required for installation:

Course name
Number of 512-byte course material blocks required
Number of auxiliary blocks required
Number of blocks required for each additional student

The procedure for *Course off* tape format will be shown.

## Copying the Material to Disk

1. Run the following BATCH job to deblock/copy the course material from the tape to a disk file.

```
/FILE 1 TAPE VOL(COURSE) RECFM(U) SHR
/FILE 2 UDS(CODEICSR) NEW LRECL(512) VOL(MUSIC1) NREC(n)
/INC DBLOCK
SELECT=x
```

   where *n* is the number of 512-byte records to be moved from the tape to disk. *x* is the file number (on the tape) where the course is located.

# Allocating the Course File

If the number of course blocks, (student and auxiliary if using *Multiple Write Access*) is less than 3900, then a file can be utilized. Otherwise, a UDS file will be needed. UDS files can handle the largest course file usable with IIPS. MUSIC requires that course files be pre-allocated and initialized. The following examples, run from a MUSIC terminal, allocate and format a course file and a UDS course file. Note that when UDS files are used the parameter BUFNO(2) is always specified on the /FILE statements.

**Course File**

```
---> /EXEC IIS.INIT
     PROGRAM TO INITIALIZE IIS COURSE AND UTILITY FILES
     ENTER  S  IF STUDENT RECORDING FILE
            M  IF MESSAGE FILE
            C  IF COURSE FILE
            L  IF LOCATION TABLE FILE
---> C
     ENTER NUMBER OF COURSE BLOCKS NEEDED
---> 2500
     ENTER FILE NAME
---> CODE:TEST.CBT
     ENTER THE NUMBER 1 IF THE FILE IS TO BE PUBLIC WRITABLE
     ENTER THE NUMBER 0 IF NOT
**** enter in 1 if you will be using "Multiple Write Access" ****
**** otherwise enter 0 ****
---> 0
     IIS COURSE FILE CREATED
```

**UDS Course File**

Execute a file containing the following:

```
/FILE 1 UDS(dsname) LRECL(512) NREC(n) VOL(MUSIC1) NEW
/INC IIS.INIT
```

In the above, *n* is the number of course blocks to be allocated. The *dsname* will be the file name. The 5th character MUST be a "#" character if the file is to be writable from any user code (Multiple Write Access).

```
      PROGRAM TO INITIALIZE IIS COURSE AND UTILITY FILES
      ENTER  S  IF STUDENT RECORDING FILE
             M  IF MESSAGE FILE
             C  IF COURSE FILE
             L  IF LOCATION TABLE FILE
----> C
      ENTER THE NUMBER 1 TO USE THE /FILE 1 FOR COURSE FILE
----> 1
      FILE CONTAINS xxxxx COURSE BLOCKS
      IIS COURSE FILE CREATED
```

Course files are defined to IIPS using the MUSIC /FILE statement. The DDNAMES of DISK20 through DISK59 MUST be used for IIPS courseware. For example, the /FILE statements for the above courses (using DISK30 as the DDNAME) would be:

```
/FILE DISK30 NAME(CODE:TEST.CBT) SHR   (no write access)
/FILE DISK30 NAME(CODE:TEST.CBT) WSHR  (multiple write access)
```

For the UDS course file, assume a *dsname* of I30.

```
/FILE DISK30 UDS(codeII30) SHR VOL(MUSIC1) BUFNO(2) (no write access)
/FILE DISK30 UDS(code#I30) WSHR VOL(MUSIC1) BUFNO(2) (write access)
```

In the above, *code* refers to the MUSIC four character code under which the course file was created and saved.

## Performing the COURSE ON Operation

1. Depending on how the course file was allocated above, add the following statements to the file $IIS:LEARN and save it as a new file called MYLEARN.

```
/FILE WORK11 UDS(CODEICSR) VOL(MUSIC1) BUFNO(2) SHR
/FILE DISK30 NAME(CODE:TEST.CBT) OLD
      or
/FILE DISK30 UDS(CODEII30) VOL(MUSIC1) BUFNO(2) OLD
      or
/FILE DISK30 UDS(CODE#I30) VOL(MUSIC1) BUFNO(2) OLD
```

2. Execute the file MYLEARN to and sign to the instructional system as the system administrator and perform the *COURSE ON* operation. Most courses will have instructions regarding the parameters to be specified on the COURSE ON commands. The following asks for 10 blank course material blocks and space for 3 students.

```
---> /EXEC MYLEARN
     ENTER STD#/CNAME OR LIST OR HELP
---> xuper
     TYPE COMMAND
---> course on 11/30
     ENTER AUTHORITY
---> iis
     TYPE CNAME/CSMATL/STUDRCDS/AUXBLKS/LAB-RATIO/HF/NEWNAME...
---> sample/10/3
     COURSE CHANGES
     CSMATL   STUDRECS     AUXBLKS    LAB-RATIO
       10         3
      TYPE OK IF CORRECT
---> ok
     TYPE CNAME/CSMATL/STUDRCDS/AUXBLKS/LAB-RATIO/HF/NEWNAME...
---> end
     END OF COURSE ON PROCESS
     TYPE COMMAND
---> register student
     TYPE CNAME/STD#/TR/P
---> sample/student
     TYPE STD NAME/LOC#/AREA#
---> student/000/00
     TYPE STD NAME/LOC#/AREA#
---> end
     TYPE COMMAND
---> sign off
```

   The last step registers the special code of STUDENT. This will allow users of IIPS under MUSIC to take the course "SAMPLE" without being individually registered. If additional students are to be registered for the course, then add them at this time.

   The dataset created in step 1 can now be deleted. The following job will remove the dataset from the system:

```
/FILE 1 UDS(CODEICSR) VOL(MUSIC1) DELETE
/LOAD IEFBR
```

   Make sure that the /FILE statement pointing to this file is also removed from your MYLEARN file. If

you will be running the course "SAMPLE" in *No Write Access*, change the /FILE statement for DISK30 to SHR from OLD.

Refer to the *IIPS/IIAS Administrator's Guide* for further details on the *COURSE ON* procedure.

## Adding New Functions

User courseware is sometimes provided with a set of user subroutines that must be installed along with the actual course. The following example will show a method of adding those *User Functions* to an IIPS system running with MUSIC. The courseware used in this example is COMSKL (IUP 5796-BBT).

1. You must restore some IIPS/IIAS macros and CMS macros. The macros will be used when assembling the *User Functions* and IIPS modules.

   Restore IIPS/AS macros from the IIPS/AS tapes.

   ```
   /INC IIS.GETMAC
   ```

   Restore CMS macros from the MUSIC source tape.

   ```
   /FILE 1 TAPE VOL(SOURCE) LR(80) BLK(13440) SHR
   /INC MFREST
   NAME='$MCC:+',REPL=TRUE,TAPFIL=2
   ```

2. On the COMSKL distribution tape, the *User Functions* are located in file three. File three is in CMS tape format and can be retrieved using the following job.

   ```
   /FILE 1 TAPE VOL(COMSKL) RECFM(U) SHR
   /INC CMSTAPE
   TAPFIL=3
   ```

   The *user Functions* will be restored to the system with the names "code:xxxxx.ASSEMBLE" where *code* is the sign-on code used for the job and *xxxxx* is the name of the function.

3. Each of the *User Functions* will need to be assembled. Execute a file containing the following job for that purpose. Remember that each of the *User Functions* MUST be assembled. In the case of COMSKL, there are seven.

   ```
   /FILE SYSPUNCH NAME($IIS:xxxxx.OBJ) NEW(REPL)
   /INC $IIS:IIS.ASM
   /OPT DECK
   /INC code:xxxxx.ASSEMBLE
   ```

   Where "code:xxxxx.ASSEMBLE" is the name of the file containing the function to be assembled.

4. Restore from the IIPS installation tape the module FUNCTION.

   ```
   /FILE 1 TAPE VOL(IIPS) BLK(8000) LR(80) SHR
   /INC GENSAV
   FILE=10,PREFIX='$IIS:',SUFFIX='.S',SELECT=TRUE
   FUNCTION
   ```

5. The *User Functions* must be known by IIPS. The module FUNCTION is used for this purpose. Refer

to the *IIPS/IIAS Operations Guide* (SH20-2459) for details. You will need to edit the file $IIS:FUNCTION.S and add the required statements.

```
Edit  $IIS:FUNCTION.S
L *PLEASE INSERT

   FUNID  NUMBER=3B0,NAME=STDATA
   FUNID  NUMBER=3C4,NAME=FIND
   FUNID  NUMBER=3C5,NAME=CHANGE
   FUNID  NUMBER=3CA,NAME=CLEAR
   FUNID  NUMBER=3CB,NAME=PLOT
   FUNID  NUMBER=3D8,NAME=CLEARF
   FUNID  NUMBER=3F8,NAME=MODIFY
```

Save the modified file for the next step.

6. Now re-assemble the FUNCTION file by executing a file containg the following:

```
/FILE SYSPUNCH NAME($IIS:FUNCTION#A.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
=FUNCTION
```

The object deck will be saved under the file name pointed to by the SYSPUNCH statement.

7. A table in the module IISSUB must be modified to reference the new functions. This format of the table is described in the module. The following shows the editor being used to apply the changes for the COMSKL functions.

```
EDIT $IIS:IISSUB.S
L SUBTAB
I        DC    X'03B0',X'0000',V(DITCW3B0),V(DITCW3B0)
I        DC    X'03CA',X'0000',V(DITCW3CA),V(DITCW3CA)
I        DC    X'03CB',X'0000',V(DITCW3CB),V(DITCW3CB)
I        DC    X'03C4',X'0000',V(DITCW3C4),V(DITCW3C4)
I        DC    X'03C5',X'0000',V(DITCW3C5),V(DITCW3C5)
I        DC    X'03D8',X'0000',V(DITCW3D8),V(DITCW3D8)
I        DC    X'03F8',X'0000',V(DITCW3F8),V(DITCW3F8)
SAVE $IIS:IISSUB#A.S
```

8. Assemble the updated IISSUB module by executing a file containing the following:

```
/FILE SYSPUNCH NAME($IIS:IISSUB#A.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
=IISSUB#A
```

9. To incorporate all of the *User Functions*, updates to the FUNCTION module and IISMUS module, a linkedit is needed. To create the linkedit job do the following.

```
EDIT $IIS:GEN.LKED
L FUNCTION
C/FUNCTION/FUNCTION#A/
L IISSUB
C/IISSUB/IISSUB#A/

      *  at this point you would include the names of the
      *  "user functions" that were assembled in step 8.
      *  In the case of COMSKL the statements would be

I /INC $IIS:DITCW3B0.OBJ
I /INC $IIS:DITCW3CA.OBJ
I /INC $IIS:DITCW3CB.OBJ
I /INC $IIS:DITCW3C4.OBJ
I /INC $IIS:DITCW3C5.OBJ
I /INC $IIS:DITCW3D8.OBJ
I /INC $IIS:DITCW3F8.OBJ
SAVE $IIS:MYLKED
```

10. Execute the job created in step 9. This will build a new IIS load module which includes the *User Functions* and updated modules.

## Adding New Screen Formats

1. The first step is to restore the assembler source or object decks for the new formats from the distribution tape. One of the MUSIC utilities mentioned earlier can be used for this purpose, depending on the method used to store the files on tape.

2. If the formats are distributed in source form each one must be assembled as follows.

```
/FILE SYSPUNCH N(format.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
/INC format
```

   where *format* is the name of the save library containing the source for the format.

3. The formats must be placed in the format library using the FORMLIB utility.

```
/INC FORMLIB
-
- the object decks for any new screen formats.
-
```

4. Once the formats have been placed in the format library the object decks or source can be removed from the system.

# Chapter 14. FSI Configuration

## Tailoring FSI Compilers/Processors Menu

The "Compilers/Processors Menu" of FSI gives users access to languages and programming systems that are installed on your MUSIC system. When you install a new product from the Optional Products tape, this menu file is automatically updated to make the new product available to your users through FSI. The menu can also be invoked directly using the XCOMPILE exec. ($INT:XCOMPILE).

You can also add items to this menu by adding items to the file $INT:COMPILERS using the editor. The following shows an example of the contents of the file $INT:COMPILERS.

```
*         JCL     Opt Description
*         Name
*
-         ASM      V  VS Assembler
-         VSF      V  VS Fortran Compiler
-         FID      N  VS Fortran Interactive Debug
-         VSC      V  VS/COBOL Compiler
-         GPS      V  GPSS V
-         VSP      V  VS PASCAL
-         PLI      V  PL/I Optimizer
-         PLT      N  PL/I Interactive Test
-         FG1      V  Fortran G1 Compiler
COBTEST   NONE     N  COBOL II Test Environment
VSAPL     NONE     N  VSAPL
SAS       NONE     N  SAS Version 5.18
PASCAL    NONE     N  Waterloo Pascal
-         RPG      V  RPG II
-         RPA      V  RPG II Autoreport
-         CB2      V  COBOL II Compiler
GDDM      NONE     N  GDDM Interface
IBMBASIC  NONE     N  IBM Basic
-         C37      V  IBM C/370 Compiler
-         R37      V  SAA RPG/370 Compiler & Lib
```

Each record contains four fields. If the first character is a "*", the entire record is treated as a comment. Records that are not comments are displayed on the menu in the order that they appear in the file. Up to 38 non-comment items are supported.

The first field can be a program name or a "-". If it is a program name, the specified program will be executed when the user selects the item. In the example above, if the user selects the GDDM Interface the program in the file GDDM is executed.

If a "-" is specified in this field it indicates that the XCOMPILE exec will handle the process. In this case, the second field contains a pointer to sample job control statements and help files, and the third field contains optional parameters.

The forth field is the description of the program or product that appears on the user's menu.

## Changing the menu items

You can use the editor to modify the description field or change the order of existing menu items. Use the TEDIT command to preserve upper and lower case.

## Adding a program to the menu

Imagine you have written a local information retrieval program that is executed by the command INFOX and you want to make it available to users through FSI. Insert the following entry to the $INT:COMPILERS file. Note that the menu items are displayed in the order they appear in the file.

```
INFOX    NONE  V  Local Information Retrieval Program
```

## The XCOMPILE Language Processor Interface

The XCOMPILE exec has built in logic to manage the user interface to various language processors. This allows people to use these languages without knowledge of MUSIC control statements or language options and parameters. The user is automatically lead through a series of steps to edit the program, compile it, execute it and review the output. If the first field in the menu file is a "-", it indicates that the XCOMPILE exec should treat this item as a language processor and handle the user interface itself.

In this case the second field is used to locate a sample control statement file and help information on the language itself. This is, by convention, a three character identifier.

For example, with VS Fortran the identifier is VSF. This means that the file $INT:JCL.VSF contains the sample control statements for running VS Fortran, and that help on the language is under the help topic VSF. The sample control file is as follows:

```
File: $INT:JCL.VSF

/SYS REGION=1024
/LOAD VSFORT
/OPT FLAG(E),SOURCE
/INC ?.S
```

The XCOMPILE exec asks the user to enter a program name. This program name is substituted into the sample control statements anywhere a ? is found. XCOMPILE then creates the following files in the user's library.

name.VSF the control statements to compile and execute the users program

name.S the source program

name.LIST output from the compile and execution

The third field tells the XCOMPILE exec how to handle the display of output from program execution. If V is specified, all the output is saved and viewed at the end of the job. When N (none) is specified the program output is displayed as it is produced. This second case is usually desirable for interactive applications.

## Modifying Existing Interfaces

You can edit the menu file ($INT:COMPILERS) to change the item description and output viewing option. Language options and file definitions can be changed by modifying the sample control statement file.

Remember that this file contains the defaults seen by all users.

## Adding a New Language Processor

Do the following to allow the XCOMPILE exec to handle the user interface for a language.

1. Invent a 3 character identifier. For example ABC.

2. Create the help text in a file ($HLP:@GO.abc). Add an entry to the HELP topics file ($HLP:@GO.TOPICS) so that "HELP ABC" will give information on the processor.

3. Create the sample control statement file ($INT:JCL.ABC). This should have the SHR or PUBL attribute and contain question marks (?) wherever you want the user's program name substituted.

4. Add the entry to the menu file ($INT:COMPILERS).

```
-       ABC  V  The ABC Language Compiler
```

# Chapter 15. ACCESS Facility

## Overview of ACCESS

The ACCESS facility allows users to connect to applications on CMS or other MUSIC systems using the MUSIC Passthru interface. Also, users can access systems outside of MUSIC such as CMS, VSE, MVS, and other MUSIC systems. The MUSIC user is automatically connected and signed on, required files are transferred, the application is run and results are transferred back to MUSIC.

## How ACCESS works

When a user issues the ACCESS command the following occurs

1. The user's MUSIC ID is checked in the AUTHORIZATION table.
2. If UNIQUE access is specified, the user is prompted for a remote userid and a password. If SHARED access is specified, a userid and password from the SHARED IDs table is used.
3. A connection is made with the remote system and the user is signed on.
4. If automatic file transfer is selected, files are moved from MUSIC to the remote system at this point.
5. If user controlled file transfer is selected, the user is presented with the file transfer menu at this point.
6. The application is started by executing the specified EXEC. A parameter can be passed to the EXEC by specifying a parameter on the ACCESS command (ie "ACCESS accessname parameter").
7. When the application terminates it should write the termination sequence to the screen. When ACCESS recognizes this, it procedes to steps 8, 9 and 10.
8. If automatic file transfer is selected, files are moved back to MUSIC from the remote system.
9. If user controlled file transfer is selected, the user is presented with the file transfer menu at this point.
10. ACCESS terminate and returns control to the MUSIC session.

## Component Description

There are four basic components to the ACCESS facility.

### ACCESS SETUP Facility

The ACCESS SETUP facility allows the administrator to define the external systems that are available and manage which users can access these systems. This facility lets you define the system and assign an ACCESS NAME. This name is used by the users on the ACCESS command to specify which application they want to use. See below for a detailed description of this facility. The ACCESS SETUP facility is invoked from ADMIN option 4 10 12, and is also documented in the *MUSIC/SP Administrator's Guide*,

under the topic "Updating External Access Tables".

## SHARED IDS Table

The SHARED IDs table is a file that contains a list of CMS userids, a password for the userid, and an optional application name for the userid. The CMS userids are used to allow MUSIC users to log on to CMS from a MUSIC application program. Thus, all MUSIC users share the CMS userids by running a MUSIC program requiring the SHARED IDs table.

An example of a MUSIC application that uses this facility is ACCESS. ACCESS calls the SIGNON subroutine to complete the signon procedure. (See below for further details on SIGNON.) If you specified SHARED IDs when you were setting up the ACCESS control file using the ACCESS SETUP facility, ACCESS calls the SIGNON subroutine with SHARED specified as a parameter in the calling sequence. (If you don't specify SHARED, UNIQUE is used which means that the user is prompted for a CMS userid and password when the time comes to log on to CMS.) This tells SIGNON to use the SHARED IDs table and find a userid to use.

The SHARED IDs table can be used to exclusively reserve a CMS userid for an application by specifying an application name for the CMS userid, and using the same application name for the application. (The ACCESS SETUP facility is used to set an application name for the application. Also, the SIGNON subroutine takes an application name as one of its parameters.) Thus, many userids can exist in the SHARED IDs table, and different userids can be reserved for different applications. This allows you to set up the CMS accounts differently based on the application.

The SHARED IDs table is displayed/updated from ADMIN option 4 10 10, and is also documented in the *MUSIC/SP Administrator's Guide*, under the topic "Updating Shared ID Table for CMS IDs".

Example of a SHARED IDs table

```
VMUSER01 VMUSERX TCPIP
VMUSER02 VMUSERX TCPIP
VMUSER03 VMUSERX SQL
VMUSER04 VMUSERX SQL
VMUSER05 VMUSERX
```

The first two CMS userids are reserved for a MUSIC program using an application name TCPIP. The second two CMS userids are reserved for a MUSIC program using an application name SQL. The last CMS userid can be used for any MUSIC program not requiring an application name.

## ACCESS Facility

The ACCESS Facility is the user program that allows users to connect to applications on CMS or other MUSIC systems. The administrator has set up these external applications via the ACCESS SETUP facility. (See below for further details on the ACCESS SETUP facility.) The command syntax is

```
ACCESS accessname [parameter]
ACCESS ?
ACCESS
```

where

accessname is the name of the external facility that you wish to access.

parameter is an optional value that is passed to the external facility once it is started.

? displays the list of external facilities available at your site.

Brief help is displayed when ACCESS is entered without a parameter.

Examples:

1. "access telnet nic.ddn.mil" logs in to the foreign host nic.ddn.mil using the VM/SP telnet program, a TCP/IP protocol.

2. "access sql" connects the user to sql on the local VM/SP system.

## SIGNON Subroutine (REXX)

This subroutine is used by the ACCESS facility to complete the signon procedure to the application. See *Chapter 22 - System Programming* for further details on the SIGNON subroutine.

# ACCESS Setup Facility

This setup program lets you, the MUSIC administrator, define the systems that are accessible to users.

The first screen lets you define the system and assigns an ACCESS NAME. This name is used by the users on the ACCESS command to specify which application they want to use. Other items on this screen include

- type of system (MUSIC or CMS)
- name of the EXEC that executes the application
- string MUSIC uses to detect application termination
- VM Passthru node of the remote system
- whether user file transfer is allowed
- CMS T-disk space
- set an application name to match with the Shared IDs file

When you start up the ACCESS Setup facility, you should fill in the ACCESS NAME that you want to create or modify. For new facilities, the name chosen should generally be descriptive of the actual function that is using invoked. For example, if the external application being accessed is the ADA programming language, an appropriate ACCESS NAME might be "ADA".

Enter the name you wish to use in the ACCESS NAME field and press ENTER. If you had previously setup the facility with the selected name, the fields on the screen will be filled in and you may make changes to the existing information. If the name entered was not previously used, the fields will be blank, waiting for your input.

The message line (at the top of the screen), provides information on whether this is an existing or new facility name.

The following describe the options available for setting up the facility.

Command to issue

This command will be issued after the Remote system has been accessed. When this option is specified and you are accessing a VM/CMS system, the following occurs:

- CMS is IPLed but NO profile exec is started
- If requested, a temporary disk is allocated
- The command is issued, and if a parameter was specified on the ACCESS command (ie "ACCESS accessname parameter"), the command is issued with the parameter

When this option is NOT specified and you are accessing a VM/CMS system, the following occurs:

- CMS is IPLed AND the profile exec (if any) is invoked

When accessing a MUSIC/SP system, and you wish to use this option, make sure that the userid that is being used DOES NOT have an AUTOPROG specified.

This field is optional.

Return to MUSIC/SP message

This field defines the message the Access Facility searches for to know when the application has completed processing and is returning to the MUSIC/SP environment.

The information entered MUST match exactly the message produced on the remote system. For example, the message "This is a TEST" is NOT the same as the message "THIS IS A TEST". The case (upper/lower letters) does matter when the Access Facility checks the message.

This field is optional, but MUST be specified if you wish to transfer files from the Remote system back to MUSIC/SP.

One method of using the above on VM/CMS is the following.

1. Create a REXX exec on CMS that has the same name as that specified in the INITIAL COMMAND.
2. In the exec perform any command to setup and invoke the desired feature/function.
3. The last line of the exec would be: say " return to MUSIC/SP message "

Example: Initial command: MYSAMPLE
Return message: Completed Processing, Return to MUSIC/SP

```
MYSAMPLE exec
      /* Sample exec */
      ACCESS 319 P
      EXEC LOCAPPL  /* start an application */
      say ' Completed Processing, Return to MUSIC/SP'
      exit
```

System Type Defines the type of the REMOTE system. This field is required. The valid entries are:

CMS - remote system is a VM/CMS environment
MUSIC(vmid) - remote system is another MUSIC/SP system. vmid is the virtual machine name in which the remote MUSIC/SP system is running.

PVM Node Defines the VM Pass-Through node if when the Remote system is accessed via VM Pass-Through. This field is optional.

Allow File Transfer TO remote system

If this option is "YES" then the user will be prompted with a menu. The user enters the MUSIC/SP filename to be transferred, the name to be used on the Remote system and any options to be used when transferring the file. If this option is set to "NO', the user will NOT receive the menu.

Allow File Transfer from remote system

If this option is "YES" then the user will be prompted with a menu. The user enters the Remote system filename to be transferred, the name to be used on MUSIC/SP and any

options to be used when transferring the file. If this option is set to "NO', the user will NOT receive the menu.

TDISK address

When accessing a VM/CMS system, the facility can auto- matically allocated temporary disk space. This space is available during the time the user is on the system. Specify the disk address to be used when allocating the temporary space. This field is optional and may only be specified when SYSTEM TYPE is CMS.

Number of blocks

Specifies the number of blocks that should be allocated when using temporary disk space under CMS. The number of blocks will be automatically allocated on the supported disk device types. This field is required if the TDISK ADDRESS field is specified.

Access mode Specifies the CMS access mode to be associated with the temporary disk space that is allocated under CMS. The mode is a letter (A-Z).

The devices supported for allocation of temporary disk space are:
FB-512, 3380, 3375, 3350

Application name

Specifies a name that is used to match against an application name for a userid in the Shared IDs table. If the names match, the userid is used for this application. Using an application name allows you to exclusively reserve a userid or a group of userids for a given application. See ADMIN 4 10 12 to add an application name to the Shared IDs table.

Example: Assume the Shared IDs table is the following

```
VMUSER01 VMUSERXX TCPIP
VMUSER02 VMUSERXX SQL
VMUSER03 VMUSERXX
```

Also, assume we have set up our application, called TELNET, using the method to update the external access table (ie ACCESS Setup facility) and assigned it an application name of TCPIP. Then, when we run "ACCESS TELNET", only the CMS userid VMUSER01 is used for this application.

Once the first screen is filled in press PF5 to create the authorization table. By default all users have access, but you can limit access to a particular user or group of users if required. This table also specifies whether SHARED or UNIQUE access is to be used for a particular user. When SHARED is specified, one of the shared CMS userids is used. These are defined using item 10 of the ADMIN04A menu. When UNIQUE is used, each user is prompted for a remote userid and password. See the "Application name" field description above on how the name and the Shared IDs work together.

The ACCESS facility can also automatically transfer files to the remote system before the application starts and transfer any result files back to MUSIC once the application has terminated. You can set this up by pressing PF7 from the main screen. This facility should not be confused with the user controlled file transfer that can be selected from the main screen. With this user controlled option, a menu is presented before and after the execution of the application, allowing the user to transfer files.

In either case, files are sent to the remote system before the application is run and back to MUSIC after its termination. To do this ACCESS must be able to detect when the application has finished.

# How to set up an Application

Assume that you have the VM/SP TCP/IP IBM program product installed on your local VM/SP system. You can set up an application for a group of users to have access to TELNET on VM/SP using this facility. There are a number of steps involved in setting up this application.

1. Allocate the CMS userids

   Allocate the CMS userids on VM/SP. Decide whether you want the application to use TDISK space, a 191 minidisk, or shared file system pool space and set up the appropriate directory for these userids. The following is a sample VM directory for the application TELNET

```
USER VMUSER01 VMUSERXX 2M 2M G 50 ON OFF OFF ON
 ACCOUNT CFVMTST MPG
 OPTION REALTIMER ECMODE
 IPL CMS
 CONSOLE 009 3215 B
 SPOOL 00C 2540 READER B
 SPOOL 00D 2540 PUNCH B
 SPOOL 00E 1403 B
 MDISK 191 3380 641 001 VMSYS2 MW
 LINK TCPMAINT 592 192 RR
```

2. Put the CMS userids in the Shared IDs table

   Use ADMIN 4 10 10 to add the CMS userids, passwords, and application names to the Shared IDs table. In this application, TELNET, add the application name 'TCPIP' to each CMS userid and password entry that you add. This reserves these CMS userids for use only with applications that have an application name of TCPIP. You can add other applications with different application names and their own CMS userids to use for that application. You can set up a second application, say FTP, and use the same application name which would give you use of the same CMS userids. This gives you the advantage of setting up a group of CMS userids that can be shared with a group of applications.

3. Set up the connect script

   Use ADMIN 4 10 12 to set up a connect script for the application. Set "Access name" to TELNET, "Command to issue" to TELNET, "Return to MUSIC message" to LOGOFF, "System type" to CMS, and "Application name" to TCPIP. Set Access Control (F5) "ID" to '*', and "Type" to SHARED. This allows all users access to TELNET, and they will be using the CMS userids in the SHARED IDs table designated for TCPIP. On a side note, if you decide to set Access Control "Type" to UNIQUE, the user is prompted to enter a userid and password for the signon when the ACCESS application is run.

4. Test it out

   Test out the TELNET application by typing "ACCESS TELNET" at *Go.

5. Tell your users about it

   Tell your users about the ACCESS facility and what applications are available to them. Note that there is online help from *Go for the ACCESS facility, and users can find out what applications are available by typing "ACCESS ?".

# Chapter 16. Configuring MUSIC for TCP/IP

## Overview of TCP/IP

MUSIC can communicate directly with VM TCP/IP Version 2 (and higher) through the socket interface. This allows you to run internet applications such as TELNET and FTP on MUSIC.

See *Appendix D - MUSIC TCP/IP for VM TCP/IP Version 1* if you are using TCP/IP Version 1. If you intend to use TCP/IP from MUSIC please read this section carefully.

## The VM Configuration

The connection with VM's TCP/IP virtual machine is done through IUCV. An IUCV path is established for each application. Set the MAXCONN parameter on the OPTION statement in MUSIC's directory entry to allow enough connections for your MUSIC users. There is no overhead or penalty if you over estimate this number.

If you want MUSIC to be able to use the raw IP interface it must be authorized in the VM TCP/IP configuration. You can do this by adding the name of the MUSIC virtual machine to the OBEY statement in the VM TCP/IP configuration file. Note that the PING program requires this.

## The $TCP:TCPIP.CONFIG File

This file contains information specific to your site, such as your MUSIC system's internet and e-mail name, the name of the TCP/IP virtual machine, the local domain name and the IP addresses of any name resolvers. The file has sample entries and comments indicating what the various parameters are. Edit the file and make any changes that are appropriate. Each entry consists of the parameter name (starting in column one) followed by the value of that parameter. The following parameters can be set.

TCPIP — The name of the TCPIP virtual machine.

HOSTNAME — The internet host name of your computer.

EMAIL — The E-mail address of your MUSIC system.

NAMESERV — The dotted-decimal internet address of a domain name server. (A number of these may be specified)

NAMEWAIT — This is the global (system-wide) name resolver timeout period, in in seconds. The resolver will read this at name resolution time and use this value for timeouts. Increase this number if your nameservers are not responding quickly enough to your requests - decrease it if response time is good and you wish to reduce the wait for bad name timeouts.

RESTRICT_TELNET_PORTS — This is a list of ports to which telnet users are to be denied access. This prevents MUSIC's status as a trusted machine from being abused by someone spoofing an internet client.

DOMAIN The domain portion of your host name.

NEWSFEED The internet address or name of a host providing an NNTP news feed.

NEWSGROUPFILE The name of a file to contain the list of NNTP news groups.

NEWSDROP_0_POSTS
Specifies whether news groups with no postings are dropped from the news groups list. (yes/no)

NEWSGROUPNAMESIZE
Maximum length of news group name before truncation.

ORGANIZATION The name of your organization.

TIMEZONE Your time zone. (GMT, EST, MST, etc.)

TIMEZONE_GMT Your time zone given as displacement from GMT (e.g. -5)

WHOISSSERVER This entry defines the whois server that the WHOIS command will connect to by default. If you have a preference, enter the server address here; otherwise, the default will be as coded in the $TCP:WHOIS program. Note that a list of whois servers can be obtained by running $TCP:GET.WHOIS.SERVERS - it ftps the file $TCP:WHOIS_SERVERS from:

```
sipb.mit.edu:/pub/whois/whois-servers.list
```

If the address 0.0.0.0 is found here, then it is not used and the the program default is used.

GOPHERD_ALLOWED_FILES
This entry defines the MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessible you specify $*:gopher*

The default is $pub:*

GOPHERSERVER This is the name,port of the default gopher server that the MUSIC gopher client will connect to. This is only used when no other pointer (startup option, -d parm, @GOPHER.MENU) is available. If this is not present and no other option is available, then the MUSIC gopher client will connect to 'gopher.micro.umn.edu' at port 70.

GOPHERTIMEOUT This is the global gopher timeout value. It is overridden by the -tm parameter in gopher.

GOPHERD_DEFAULT_MENU
This entry defines the default FIRST MENU to be transmitted when no menu is specifically requested of the GOPHERD server. The default is $tcp:gopherd.menu

GOPHERD_BBS_LOG This entry defines a BBS log file for logging BBS topic file access. No menu is specifically requested of the GOPHERD server. This must be in the form "userid:@xxx.". Example cw99:@inf. is used for INFOMcGill. The default is none.

HTTPD_PERSONAL_PAGE
: This entry defines the default PERSONAL page to be transmitted when no page is specifically requested in the URL. for example http://musicm.mcgill.ca//abcd/ will get the default personal page text tacked on. The default is http\home.HTML

HTTPD_ALLOWED_FILES
: This entry defines the MASK for any file allowed to be accessed via HTTPD. For example, to allow the subdirectory HTTPD on userids that begin with dollar to be accessible, you specify $*:HTTPD* The default is *:*HTTP*

HTTPD_ALLOW_PUBLIC
: This entry specifies whether public files not fitting the HTTP_ALLOWED_FILES mask, should be accessible. The default is no.

HTTPD_ALLOWED_EXECS
: This entry defines the MASK for any file allowed to be executed via HTTPD. This is specifically meant for forms support in html files. The default is *:*HTTPEXEC*

HTTPD_DEFAULT_PAGE
: This entry defines the default page to be transmitted when no page is specifically requested in the HTTPD connection. The default is $000:http\home.HTML

# Domain Name Servers

MUSIC will take advantage of a Domain Name Server if one is available. A Domain Name Server is usually a small computer that is dedicated to providing name lookup services. It keeps in contact with other name servers around the network to keep its information up to date. Given a host name in the format:

```
vm.mpg.mcgill.ca
```

It can quickly return the numeric internet address that is used by the TCP/IP applications to establish connections with remote hosts. When a user specifies a host name on a TELNET of FTP command, MUSIC asks the Domain Name Server to resolve the name and return the numeric internet address. The advantage of using names instead of numbers far outweighs the small overhead in doing the name lookup.

If a name server is not available, your users will have to specify all host addresses in the "dotted" decimal format (e.g. 132.206.120.2).

Domain Name Servers are defined in the $TCP:TCPIP.CONFIG file on the NAMESERV statements. The first one should be regarded as the primary name server. It is tried first. If it fails to respond within a 15 second time out period the system will try the second name server if one is defined. Note that if the primary server is down, users will see a 15 second delay in resolving host names. Extra name servers are only defined for emergency backup. If the primary name server is going to be down for any length of time it should be removed from the top of the list so that a backup can take over without the 15 second delay.

# The $TCP:NET.LIST File

This provides your users with a list of network hosts and the services they offer. It is used by TELNET, FTP and NET. The list does not restrict or grant access to the host computers, it simple lets the user know that they exist. The sample that is distributed with the system should be modified to suite your local needs.

The format of the file is relatively simple. Each network host is represented by a record in the file. The first field of each record is the network address of the host. This can be specified in the dotted decimal format (e.g. 132.206.120.2) or in host name format (vm.mpg.mcgill.ca). The rest of the record should contain a description of the host or information about what services are available on that host.

The contents of the file are up to you. The file may simply contain a list of the computers on your campus, or could contain thousands of entries describing computers throughout the internet. At McGill for example we provide a list of archive sites that allow anonymous FTP access.

## TCP/IP Log Files and Console Messages

TCP/IP applications use the log server to generate log files. Log file names are in the format:

```
$TCP:TCPIP.yyymm
```

where *yyyy* is the year and *mm* is the month. These log files contain messages from both outbound and inbound applications. Outbound applications that log actions to these files are TELNET and FTP; inbound applications are FTPD and FINGERD (FTPD and FINGERD are described later).

Also, FTPD, FINGERD and INETD (described below) issue messages to the console. This can be useful in checking to see if there is a problem with excessive access to your system. As well, in the case of an actual attempt at system penetration, these trails are useful in tracking remote connections, and in identifying TCBs where remote users are logged on, etc.

## The Internet Super Server (INETD)

The INETD program (internet daemon) acts as an entry point for incoming connection requests to TCP/IP servers running on MUSIC. Its basic job is to wait for a call to come in on a port and then dispatch an appropriate server for that port.

In $PGM:SYSLG.DEFINE, TCP/IP applications are defined as monthly log files. INETD runs in the background as a BTRM. It creates sockets for each of the ports defined in the file $TCP:INETD.PORTS file and listens for incoming calls. When a call comes in, it accepts the connection and executes the appropriate server for that port. The server is started as another BTRM and is passed the connection. In other words, INETD runs one copy of the server for each incoming call. As the server program performs its task, INETD goes back to listening for more incoming calls.

Before INETD accepts a connection, it verifies that the incoming ip address and port are not in the killfile $TCP:IP.AUTHORIZE. If they are in the killfile, the connection is refused and a shutdown on the connection is done. If they are not in the killfile, INETD accepts the connection and it executes the appropriate server for that port.

Note that each copy of a server requires a MUSIC session. If you anticipate high usage, make sure you have sufficient extra sessions defined in the NUCGEN (XSES parameter) to handle the load.

The file $TCP:INETD.PORTS contains the list of ports and servers that are handled by INETD. The following is a sample.

```
;
;  The following are default ports
;
21   stream 10 $tcp:ftpd
```

```
70   stream  0 $tcp:gopherd
79   stream  0 $tcp:fingerd
80   stream  0 $tcp:httpd
106  stream  0 $tcp:poppassd
110  stream 10 $tcp:popd
370  stream 300 $mcs:mcsh
;
;  The following define ports for demonstration servers.
;
2000 stream  0 $tcp:demo.serv.fort
2001 stream  0 $tcp:demo.serv.rex
3000 stream  0 $tcp:demo.telnet.rex
```

The first column contains the port number of the service, the second the type of port, and the third contains the maximum number of simultaneous sessions allowed on that port (zero indicates no limit).

The remainder of the record contains the name of the execution file for the server, followed by any parameters that are to be passed. For example, if a connection comes in on stream (TCP) port 21, INETD will execute the program "$tcp:ftpd".

The port number should be between 1 and 3999. Certain ports are know as well known ports because they run well known services. For example:

```
  21 - File Transfer Protocol (FTP)
  23 - TELNET
  25 - Simple Mail Transfer Protocol (SMTP)
  53 - Domain Name Server
  70 - Gopher Server
  79 - Finger Server
  80 - Web (HTTP) Server
 110 - POPD Server
```

There are two values for port type: "dgram" and "stream". "dgram" ports are not really handled by INETD, it simply starts the server as a BTRM. In the above example INETD will start the program "$000:msgd" as a BTRM, but take no part in handling calls or data on the port.

The following files are distributed executor files for servers available with MUSIC, eligible for inclusion in the file $TCP:INETD.PORTS.

| | |
|---|---|
| $TCP:FTPD | FTP (File Transfer Protocol) server |
| $TCP:FINGERD | Finger RUIP (Remote User Information Protocol) server |
| $TCP:GOPHERD | Gopher Protocol server |
| $TCP:HTTPD | Web server |
| $MCS:MCSH | MUSIC/SP Client/Server |
| $TCP:POPD | POP (Post Office Protocol) server |
| $TCP:POPPASSD | POP Password server |
| $TCP:DEMO.SERV.FORT | demonstration server writtem in VS/FORTRAN |
| $TCP:DEMO.SERV.REX | demonstration server written in REXX |
| $TCP:DEMO.TELNET.REX | demonstration TELNET server (line mode) |

The file $TCP:IP.AUTHORIZE contains a list of ip address and port combinations which are not allowed access to the defined service. Before INETD accepts a connection, it verifies that the incoming ip address and port are not in the killfile $TCP:IP.AUTHORIZE. If they are in the killfile, the connection is refused and a shutdown on the socket is done. If they are not in the killfile, INETD accepts the connection and it executes the appropriate server for that port. The following is a sample.

```
*
*---Sample authorization file
*
*   Remove the leading  *' to activate the following statements
*
*Address          Port   Optional Comment
*-------          ----   ----------------
*vm.mpg.mcgill.ca  79    testing access via port 79
*123.456.789.012   21    numeric IP addresses also acceptable
*   *              70    any site; port 70; i.e. no gopher
*123.456.789.012   *     a particular site, any port
```

The first column contains an ip or symbolic address for which you want to block access. ";" or "*" in this column signifies a comment line. Enter a blank followed by "*" in column one if you want to restrict everyone.

The second column contains the port number to restrict or "*" to restrict all ports.

Anything after the second column is comments and can be used to document the entry.

The following options can be set as namelist parameters. Edit and modify the file $TCP:INETD to change these values.

TRACE=T/F specifies whether tracing is enabled or disabled. When tracing is enabled, tracing information from the socket calls is recorded in the file @TCPIP.LOG, and tracing information from the socket I/O is recorded in the file @TCPIP.BUFFERS. The default is TRACE=F.

RESET=T/F specifies whether INETD restarts (true) or halts (false) after ERRLIM errors are encountered. The default is RESET=T.

ERRLIM=nnnn specifies the number of consecutive SELECT or ACCEPT errors are accepted before INETD is stopped. INETD can be automatically restarted when it stops by setting RESET=T. The default is ERRLIM=20.

RETRYD=nnnn specifies the seconds to delay when a SELECT error occurs. The default is RETRYD=10.

RETRYB=T/F specifies whether BIND failures are retried every BINDTM seconds for BINDCT retries or whether BIND failures are not retried. The default is RETRYB=T.

BINDTM=nnnn specifies the seconds to delay between BINDCT retries when a BIND failure occurs. The default is BINDTM=31.

BINDCT=nnnn specifies the retries to attempt when a BIND failure occurs. The default is BINDCT=20.

TSKTIM=nnnn specifies the seconds waited for a server task to be started. If the server has not started, INETD does a shutdown on the socket. The default is TSKTIM=30.

NOAREP=T/F specifies whether the attempted access violations are reported (false) or not reported (true). Access is limited by use of the file $TCP:IP.AUTHORIZE. The default is NOAREP=F.

**To use INETD:**

1. Add a BTRM definition to the NUCGEN to run the program. Also make sure you allocate enough extra sessions (XSES parameter) to allow for the additional server sessions.

2. Create a $MONxxx userid with INETD as the autoprogram. This userid must be given the privileges

required to run any of the servers. The FTP server requires SYSCOM, MAINT, FILES, CODES, LSCAN, CREAD, VIP, and JOBLIM. Other servers may require other privileges.

```
(xxx-the device address of the BTRM)
```

Example of CODUPD command:

```
ADD $MON SC(xxx) AUTOPROG($TCP:INETD) NAME(BTRM FOR INETD) -
PW(*NOLOGON) BAT(0) PRIME(NL) NONPRIME(NL) DEF(NL) XSESS(200) -
SYSCOM MAINT FILES CODES LSCAN CREAD VIP JOBLIM
```

3. Add entries to $TCP:INETD.PORTS for any servers you wish to run.

# FTP Client (FTP)

The FTP client is used to conduct FTP sessions with remote servers. There are a number of parameters available to configure or tailor your FTP client on a system-wide basis. (Individual users can of course override these defaults.)

The following options can be set as name-list parameters. To change these, edit and modify the file $TCP:FTP.

| | |
|---|---|
| BUFSIZ=n | specifies the amount of buffer space used in TCP/IP requests - this number has an impact on the system buffer pool (default 8k, min. 50 bytes, max 31744 bytes). |
| WAIT=n | specifies the timeout period |
| ATODIR=T/F | specifies whether automatic directory display is enabled or not (default: autodir is enabled) |
| DIRIN=T/F | specifies whether or not to use LIST command by default (default: LIST is used instead of NLST) |
| DEFSP=n | specifies the default primary space allocation for created files (default: 100k) |
| TRA2E='name' | specifies the replacement ASCII-EBCDIC translate table (default: no replacement) |
| TRE2A='name' | specifies the replacement EBCDIC-ASCII translate table (default: no replacement) |
| TRE2E='name' | specifies the replacement EBCDIC translate table for screen display |
| FS=T/F | controls whether fullscreen startup is enabled or note (default: fullscreen startup is enabled) |
| SUNIQ=T/F | specifies whether or not to use STOU for FTP PUT requests instead of STOR (default: STOU is used) |
| LOG=T/F | specifies whether a log file entry is written (default: log file entries are written) |
| DEBUG=T/F | same as -t at startup - turns tracing on. This is useful for a standard debugging executor. (Default: this is not enabled.) |
| BINXFR=T/F | specifies whether the MUSIC FTP client is to attempt BINARY BLOCK transfers when connected to a MUSIC server (default: this is enabled). |

# FTP Server (FTPD)

The FTPD program works with INETD to provide the FTP server function. If you wish to provide this service you must activate INETD (see the previous section). It is also possible to start FTPD on its own. This will support only one connection at a time and is intended for debugging purposes.

The FTPD program has the following options that can be set as name-list parameters. To change these edit the file $TCP:FTPD.

ANONID='anon1 ,'anon2',... Userids that can be used to gain anonymous access to FTP. The default is ANONYMOUS. These users will be allowed to access files belonging to the MUSIC userids specified in the positionally corresponding PUBCOD parameter. Up to 20 anonymous userids can be specified.

ANONOK=T/F This parameter determines whether anonymous access is allowed. The default is T (true). If you wish to disable anonymous access set this to F (false).

DROP=n The number of minutes to wait before dropping an idle connection. The default is 10 minutes.

PORT=n The port number that FTPD should listen on for incoming connections. This is NOT used if FTPD is started by INETD. It is used when you start FTPD on its own for debugging purposes. Note that this SHOULD be different from the port that INETD uses - TCP/IP will accept 2 or more servers running on the same port number, and pass calls to them each in turn (in a round robin fashion).

PUBCOD='userid'...'userid'
A list of MUSIC userids associated with the anonymous userids as determined by the ANONID parameter above. These need not be different, but you should note that these are effectively public, and the information on them should be treated as such. The default is $PUB. The maximum is 20 userids.

UPANON=T/F,T/F,T/F
This determines whether anonymous users are allowed to upload files to your public library. The default is F (false). The maximum is 20 values (corresponding to the maximum 20 anonymous userids).

TRFLAG=T/F This is used to activate tracing. The trace information is written to unit 6 (the screen) but can be mapped to a file if FTPD is run in the background. This is intended only for debugging and the default is F (false).

## FTP Ports

According to TCP/IP standards, port 21 is reserved for FTP services. By default VM TCP/IP uses port 21 for the CMS FTP server. If you wish to continue to provide this access, MUSIC's FTP will have to use another port, perhaps 1021. In this case the entry in the $TCP:INETD.PORTS file would read:

```
1021 stream 10 $tcp:ftpd
```

Port 21 would give FTP access to the CMS file system. Port 1021 would give FTP access to the MUSIC file system. Users would have to specify port 1021 on the FTP command to access MUSIC. If most of the files of interest are on MUSIC you may wish to give MUSIC the default FTP port. In this case the entry in $TCP:INETD.PORTS would be:

```
  21 stream 10 $tcp:ftpd
```

In addition to this you would have to modify the PORT section in "TCPIP CONFIG" on CMS and change PARM option in the FTPDEXIT EXEC to prevent the CMS FTP server from taking port 21 for its own use. (See the section on "Configuring the FTP Server" in the Planning and Customization manual for VM TCP/IP.)

## Multiple FTP Servers

You may also wish to run a number of different FTP servers on MUSIC. Perhaps one for general use and another to give anonymous access to a specific group of files. In this case the $TCP:INETD.PORTS would contain an entry for each server:

```
  21 stream 10 $tcp:ftpd
2021 stream  5 $000:ftpd2
```

Calls coming in on port 21 would get the standard server, those on port 2021 would get a modified server executed from the file $000:ftpd2.

# FTPD Security Exits

These exits allow a site to tailor the behaviour of the FTPD server with respect to allowing and disallowing incoming calls. INETD already allows a site to control the number of concurrent sessions being run by a server process; the FTPD server is not even invoked at that point. These exits provide a means of fine-tuning access based on time-of-day, remote site id, etc. For example, during the daytime hours, only local FTP connections might be allowed, or only certain FTP commands might be allowed.

There are four exits defined. They are:

| | |
|---|---|
| EXCODE | provides for validation of the userid/password entered via the USER and PASS FTP commands |
| EXCONN | provides for validation of the remote IP address |
| EXCOMM | provides for validation of any command entered by the remote user |
| EXQUIT | called when the connection is shutting down, for logging purposes |

In order to install the exit procedures, you must do the following:

1. Replace the module $TCP:FTPD.$EXITS.S with your routines. You can follow the model there (it uses ENTRY statements within one routine) or you can provide separate routines. You should take care not to use COMMON blocks from the main program, as in future these exits will be installed in the load library, and such information will not be available. The names EXCONN, EXCODE, EXCOMM, and EXQUIT must be resolved at link-edit time. Keep in mind while coding your routines that the module FTPD is not run as a privileged program (FTPD has no privileges) but rather is run on a privileged userid.

2. Compile your routines into the file $TCP:FTPD.$EXITS.OBJ.

3. Review the file $TCP:FTPD.LKED. Make sure that any routines (.OBJ files) exist on disk. (Note that .OBJ files are commonly not resident - you might have to use ADMIN to restore these files.) Once satisfied, re-link FTPD by running $TCP:FTPD.LKED.

4. Review the linkage editor output. If there are no errors, then your new version of FTPD is in production. (You can test it by running it from a terminal session - you'll need SYSCOM on the test userid.)

In the future, these exit routines will be installed in the system load library, and a much system installation process will be available. In the meantime, do not make use of any common block information within the FTPD source module. Later versions of the FTPD server with linkage to the exit routines performed from the system load library will make access to this information impossible.

## Descriptions of the User Exits

### The EXCODE Exit

The EXCODE user exit is invoked from FTPD as follows:

```
CALL EXCODE(RC, USERID, ULEN, UPASS, PWLEN, PUBCOD, ANONYM)
```

Arguments:

| | |
|---|---|
| USERID | 16 character userid |
| ULEN | raw length of the userid (note that this is NOT the file ownership id) (4) |
| UPASS | the password (8 char.) |
| PWLEN | the password length (4) |
| PUBCOD | the 16 char. userid of the public directory to use. |
| ANONYM | pointer to the anonymous userid common block. This can be moved to your own common block for processing. This should not be altered. The common block looks as follows: |

```
CHARACTER*20 $PUB(20)
CHARACTER*20 ANONID(20)
LOGICAL      ANONOK
INTEGER      NANON
COMMON/ANONYM/ANONOK,NANON,ANONID,$PUB
```

*Note:* Do NOT use this common block directly. Future versions of the FTPD user exits will be loaded from the load libarary and this will not be available.

This exit provides the site with greater control over access to the FTP resource. Access can be denied based on time of day, etc. This exit is called in two places:

a. when processing the USER command, if the user enters an id that matches an anonymous userid (for which by default there is no password. Note that normal processing in this case means that the login is accepted, and that EXCODE will NOT subsequently be called with a password. If you want to block the login, this is it. RC=1 on entry here.
b. when processing the PASS command. The password that the user entered is provided. Note that FTPD runs with the CODES priv, so this exit can take advantage of that and query the code table itself. RC=0 on entry here.

When EXCODE returns, processing action taken will use the new values, if any, in USERID, ULEN, PASSWD, PLEN. These may be overriden by the processing action, in the case where you request that a password be required and it is an anonymous userid.

Meaning of the Return Codes that your routine sets:

0 - permit login (normal processing)
1 - reject login (socket is not closed)

2 - reject login (shutdown requested)
3 - allow login, but require password, even
if anonymous
4 - allow login, no password required.

Notes on Individual Cases (by return code)

rc=0 requests normal processing, as if no exit routine were called. This is the default action without an exit routine supplied.

rc=1 login is rejected, but the remote user remains connected. The user is free to try again, subject to the limit of the LOGATM value specified in the EXCONN exit. If the user has no hope of logging in due to the local time, etc. this is faster than rejecting at the PASS command.

rc=2 the userid is rejected, and the socket is shutdown. This is perhaps appropriate when login is disabled for any reason, and you wish to discourage attempts to login.

rc=3 normal processing can continue, but the user must supply a password. This applies even to anonymous login, so that this exit WILL be called again when the PASS command is processed. At this point, rc=4 can be specified to allow any password entered. This option allows the site to check for userid@site-address password forms for anonymous logins.

rc=4 Login is allowed, and in fact this return code validates the user. No subsequent access to the code table is required. This can be used in conjunction with rc=3 to enhance anonymous access by requiring a user@site format password string, which is not passed to the code code table.

Note that if login is denied, then message 531 is returned to the remote client.

**The EXCONN Exit**

The EXCODE user exit is invoked from FTPD as follows:

```
CALL EXCONN(RC, RMTIP, RMTPRT, LOCIP, LOCPRT, DROPTM, LOGATM)
```

Arguments:

| | |
|---|---|
| RMTIP | Remote IP Address |
| RMTPRT | Remote Port |
| LOCIP | Local IP Address |
| LOCPRT | Local Port |
| DROPTM | Drop time in seconds (modifiable) |
| LOGATM | Number of login attempts (modifiable) |

These values are passed by value only except as noted. All of these are of type INTEGER*4.

Meaning of the Return Codes that your routine sets:

| | |
|---|---|
| 0 | - normal processing (call is accepted) |
| nonzero | - a message of 430 is issued and the socket is shutdown. |

This exit allows you to ascertain if you wish the remote connection to be continued. The information provided is read-only - you cannot change it. It is provided only so that your user exit can decide whether to shutdown the connection. Note that you do not have access to the socket - if you wish to issue a message, a non-zero return code must be returned. Halting the program via CALL EXIT will also have the same effect,

except that VM's TCP/IP will issue a message that the connection was broken. With the 430 message, the user knows why the shutdown occurred.

#### The EXCOMM Exit

The EXCOMM exit is called from the FTPD server as follows:

```
CALL EXCOMM(RC, CMD, INPUT,RLEN, CMD,CLEN, CMDOPS,OPLEN, NXTCMD)
```

Arguments:

| | |
|---|---|
| INPUT | raw message from the remote user |
| RLEN | length of this message - do not exceed this |
| CMD | Command as parsed by FTPD |
| CLEN | length of the command |
| CMDOPS | command operand(s) as parsed by FTPD |
| OPLEN | length of the operand field |
| NXTCMD | the next command that FTPD is looking for. (eg. user/PASS, rnfr/RNTO, etc.) Set this to blank to cause FTPD to ignore this test. |

Meaning of the Return Codes that your routine sets:

0 - continue normal processing
1 - go directly to command interpretation
2 - Abort command: put out a message and get next user command
<0 - FTP server message code - see below

This allows your user exit to determine what will happen as a response to the user command just issued. If RC is negative (RC < 0), then the positive value of RC is interpreted as an FTP message code which is issued to the user. In this case, CMD is the (optional) text used to tailor the message. If no tailoring text is desired, set this field CMD to blanks. If R=2, then message 505 is issued, aborting the command. RC=1 means bypass any further command validation and execute the command.

#### The EXQUIT Exit

No optional actions are provided for. EXQUIT gains control only for local purposes (logging, etc.). Thus, no return code is required and no parameters are passed.

# POP3 and POPPASS Servers on MUSIC

## Introduction to POP

Post Office Protocol, POP, is a client/server model for mail designed to work with TCP/IP. The user can run POP client software on any platform, connect to a POP server, and download their mail to the client. The POP client software acts as the Mail User Agent, MUA. All clients can display the mail and perform various mail handling tasks such as answering it or copying it to a file. Clients also allow you to create mail to be sent without being connected to the server (ie offline), and then connecting to the server immediately or later and sending the mail. The POP server software acts as the mail Message Transport System, MTS. The POP server platform has sufficient resources for this service, whereas the POP client platform typically does not have the resources for this task. The POP client software allows the user to work offline and connect to the POP server only when required.

The POP specifications are developed by the Internet community and its governing bodies. The latest version of POP is POP version 3, POP3, and it has been widely implemented on most computer platforms.

MUSIC users have a choice of how they view their mail. Users can use the traditional way of connecting to MUSIC via a tn3270 client or a direct connection and view their mail interactively with the MUSIC MAIL program. Alternatively, users can use a TCP/IP POP3 client on their own computer, download their mail from MUSIC via the MUSIC POP3 server and view their mail on their computer.

## POP3 Server

The MUSIC POP3 (Post Office Protocol version 3) server is a TCP/IP server that allows a user using a POP3 client to download their mail from MUSIC. Your users can connect to MUSIC with a POP3 mail client such as WinPMail or Eudora on a PC, or Eudora on a MAC, and fetch their mail from MUSIC. Once they have fetched their mail, they can read it offline. The POP3 clients allow the creation of mail to be sent offline and they have the ability to send it immediately or at a later time. Use of a POP3 client reduces the connect time to MUSIC when you compare it to the connect time for a user who signs on to MUSIC via tn3270 and reads their mail interactively.

Mail attachments and MIME support are available to your users via the MUSIC POP3 server as these features are a function of the POP3 client. Most POP3 clients include support to allow the attachment of any file(s) from the local environment to the message being sent. So, a user can send their WordPerfect, Microsoft Word, or spreadsheet file via mail to another person. Also, most POP3 clients support MIME, Multipurpose Internet Mail Extensions. MIME provides facilities to include multiple objects of any type in a single message. For example, a mail message can contain a text object, an image object, and an audio object. When the POP3 client receives a MIME message, usually a user can choose which object to "view" in the message.

## POPPASS Server

The POPPASS server is a TCP/IP server that allows a user to change the password for their POP3 userid. On MUSIC/SP, the user changes their signon password as the POP userid is their signon password. This server supports the Eudora POP client "Change Password" and the WinPMail POP client "Change POP3 Password" requests. The Eudora products for various platforms are available from QUALCOMM Incorporated. The WinPMail product is available free on the internet.

## POP3 Commands and RFCs

The MUSIC/SP POP3 server complies with RFC1939 and supports the optional POP3 commands:

| | |
|---|---|
| TOP msg n | returns the mail headers and "n" lines of the mail body in a multi-line response. |
| UIDL [msg] | returns the unique-id of the message. |
| APOP name digest | is an authentication mechanism that does not send a server/userid specific password in the clear on the network. |
| USER name<br>PASS string | when USER and PASS commands are used together is an authentication mechanism that sends a server/userid specific password in the clear on the network. |

RFC1939 is the latest revision (May 1996) of Post Office Protocol Version 3 and it is a Standard Protocol.

The MUSIC/SP POP3 server also supports RFC1725 and RFC1460 which RFC1939 obsoletes, allowing

POP3 clients that are only RFC1725 and/or RFC1460 compliant to work with the server.

## XTND XMIT Support

The MUSIC POP3 server also supports the Berkeley extension XTND XMIT which allows the POP3 client to send mail to the POP3 server for delivery. If the server does not support this feature, the POP3 client must send the mail to a SMTP server. The MUSIC XTND XMIT support includes sender verification. This prevents a user from masquerading as another user by setting a "sender" header in the mail to someone else's email address. The MUSIC POP3 server verified that the POP userid's email address is in one of the From/ Sender/Reply-To "sender" fields. If it is not, the POP3 server refuses to send the mail on the user's behalf and signals an error to the client.

## Mail Management

Mail management within the POP3 server mail store is accomplished by the same features that are included in the standard MUSIC system. Sites can tailor the retention time of mail in the store for each user, group of users, or class of users. An automatic mail cleanup job can be set up to delete the expired mail from the system on a regular basis.

## POP3 Clients

The POP3 server has been tested with WinPMAIL 2.01 (freeware), Eudora 2.0.3 for Windows (commercial software), Eudora 1.4 (freeware), and Spry's Air for Windows AirMail v3.0 (commercial software) and higher level versions working with the Novell LanWorkplace Winsock and Microsoft's Windows 95 TCPIP support. It has also been tested with Microsoft/Exchange Windows 95 Version 4 Internet mail option. The POPPASS server has been tested with Eudora 2.0.3 for Windows (commercial software), and Eudora 1.4 (freeware) and higher level versions, and WinPMail v2.54 (freeware) working with the Novell LanWorkplace Winsock and Microsoft's Windows 95 TCP/IP support.

The MUSIC/SP Ph server can be used for those mail clients such as Eudora 3.0 (32) for Windows (commercial software) and WinPMail v2.54 (freeware) that support Ph database queries. This version of Eudora supports this feature via its Directory Services option. WinPMail supports this feature via its Extentions option. See the file $TCP:PH.DOC for further details on the MUSIC/SP Ph server.

### Latest POP3 Documents

To see the POP3 server documents coming from MUSIC or for details of how to obtain the free POP3 clients, just point your Internet Web browser to:

```
http://MUSICM.McGill.CA/msi/http/pop.html
```

### Trademarks

## POP3 Server Installation

As the POP3 specifications evolve, updates to the POP3 server are made available by anonymous FTP to MUSICM.MCGILL.CA. The following notes outline install procedures:

- If you have not installed the MUSIC TCP/IP support yet then you must do so first. Refer to the section "The Internet Super Server (INETD)" in this manual.

- If you are going to run the POPPASS server, you should add all of the code privileges to the BTRM running $TCP:INETD. These privileges are documented in this manual under the CODUPD program. These privileges are:

  ```
  SUPV LSCAN SYSCOM CREAD INFO FILES SYSMNT CODES
  DREAD MAINT VIP JOBLIM PRIV12 PRIV13 PRIV14 PRIV15
  ```

  Since a CODXXX job is run to change the password, if the BTRM userid does not have the same userid privileges as the userid whose password the BTRM is changing, the password change will fail. This is a standard action for all CODUPD/CODXXX programs. Giving the BTRM all the privileges allows the BTRM to change the password for any userid.

- Run MAIL.CONFIG to configure $TCP:POPD and $TCP:POPPASSD, and set the program parameters to agree with your current MAIL.CONFIG settings. See the Configuration Notes below for further details on the namelist parameters.

- If you have logging on for your MAIL facility in MAIL.CONFIG and for TCPIP, you should add the following entries to the system log server BTRM define file, $PGM:SYSLG.DEFINE:

  ```
  MAIL  TO MAIL1
  TCPIP TO TCPIP1
  BADPW TO BADPW
  MAIL1:  FILE($EMD:@MAILLOG) SP(100) SECSP(100%) DAILY SHARE
  TCPIP1: FILE($TCP:TCPIP.) SP(100) SECSP(100%) MONTHLY SHARE
  BADPW: FILE($000:BADPWLOG) SP(50) SECSP(100%) MONTHLY SHARE
  ```

  The POP3 and POPPASS servers both log failed signon attempts because of a bad password or userid to the application name BADPW. This is provided for the site to monitor attempts to break into the system. You can turn off logging (LOG=F) if this is not important for your site.

- Edit $TCP:INETD.PORTS and add lines at the bottom that reads:

  ```
  106  stream  10 $tcp:poppassd
  110  stream  10 $tcp:popd
  ```

  These definitions use the standard ports 106 and 110 for the POP3 and POPPASS servers respectively. These definitions allow 10 clients at one time for each server.

- Either re-ipl your MUSIC system or issue a /RESET on the BTRM that runs INETD. (You will see this number on the console messages at startup time and on the console messages associated with each GOPHER or FTP connection.)

## Usage Notes

- The POP concept is limited in that you download your mail to another machine where it may be stored and delete the mail from the server. If you access a number of machines to read your mail, for example at the office and at home, the mail is not available in both environments. POP3 clients work around this

functionality issue with a feature that allows you to read your mail from any number of machines, but then the mail must not be deleted from the server. This issue puts the reliance on the system to manage the mail on the server.

Most POP3 clients allow the user to set an option that will leave the mail on the server, and not delete it once it is downloaded. For example, Eudora's option "Leave mail on server" is found in the Special item under Switches. Note that once mail is read on the server, the client usually decides not to download "old" mail. It only downloads new (ie unread) mail.

MUSIC/SP offers an ideal solution to the "mail must be deleted from the server" issue. With the combination of default expiry dates for user mail, the POP3 server's expired mail removal feature, and the mail cleanup program, the mail is managed on the server, MUSIC/SP.

- The MUSIC/SP POP3 server supports the RFC1460 POP3 LAST command between sessions. This is a more efficient way for the client to locate the first unread message in the mailbox on the server. This means that any mail previously read will not be downloaded to the client when the client has been told to use the POP LAST command. For example, Eudora supports the LAST command with the UsePOPLAST=1 entry in the [Switches] section of the file EUDORA.INI.

- The MUSIC/SP POP3 server uses the non-standard RFC822 header Status: R to flag old mail. Some POP3 clients use this information and do not download the "old" mail, only new mail. The POP3 server works hand in hand with these clients.

- The MUSIC/SP POP3 server creates RFC822 headers for SENDFILE items before they are downloaded to the client. By putting RFC822 headers on the SENDFILE mail item and thus "converting" it to a regular mail item, clients can now display the item's subject, sender, and send date.

- Your client software may support the send operation by connecting either to the defined SMTP server and using the SMTP protocol, or by using the POP3 server using the POP3 XTND XMIT feature. The commercial version of the Eudora mail client supports the POP3 XTND XMIT feature. For Eudora, under the tools options menu, sending mail category, the field "SMTP server" determines whether mail is sent via SMTP or POP3 XTND XMIT. Defining the SMTP server address sends mail via SMTP. When this address is left blank, Eudora sends the mail via POP3 XTND XMIT. Earlier versions of the commercial version of Eudora supported XTND XMIT when the line UsePOPSend=1 was found in the file EUDORA.INI in the [Switches] section. UsePOPSend=0 sent mail via SMTP. XTND XMIT mail sending is the faster method.

- Your site may have configured the POP3 server to restrict your access to your mailbox via your pop mail client. If this is the case, you may see the message "MAILBOX access denied, retry in x minutes" if your pop mail client reflects messages back to you. When you get this message, wait the indicated minutes before attempting to reconnect to the POP3 server.

## Configuration Notes

The Mail facility configuration program $EML:MAIL.CONFIG can be used to set the common namelist parameters for the POPD and POPPASS servers, and generate the execs for the servers. Descriptions of the various namelist parameters are given below.

#### $TCP:POPD namelist parameters

The namelist parameters for $TCP:POPD are MUSNOD, ALIASn, N$xx, ZONE, LOG, CONSOL, ACCESS, ERRFIL, PCODE, PSTMST, DEADX, SNOOP, CODTBL, DBCS, EXPIRE, PORT, BUFSIZ, DROP, TRACE, APOP, POPFRQ, and REG. All of these parameters except TRACE are the same parameters as defined in $EML:MAIL.CONFIG for the Mail facility. The definitions for the important namelist

parameters PORT, BUFSIZ, DROP, TRACE, APOP, POPFRQ, and REG are given in $EML:MAIL.CONFIG and below.

PORT=nnn Where nnn is the port number to run this server on. PORT=110 is the default. 110 is the standard port for POP3. If run interactively, the default is PORT=3110.

BUFSIZ=n Where n is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=31744.

DROP=n Where n is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value.

TRACE=t or f TRACE=t can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=f.

APOP=t or f The APOP=t parameter can be used to force the POP3 server to accept only APOP authentication from any client. APOP=f allows any authentication method (ie USER/PASS and APOP). The default is APOP=f.

POPFRQ=n Where n defines the frequency in minutes a user can access the POPD server within a one day period. POPFRQ=10 (once every 10 minutes) is the default. POPFRQ must be between 0 (no limit) and 1440 (1440 minutes). POPFRQ=0 when it is set a value outside of this range. This parameter sets the system default value for all users. It can be overridden by a setting for POPFREQ in the Mail Authorization Table type 2 record for a user or group of users.

REG=n Where n defines the region size to use for the POPD server. The default is REG=600 (i.e. 600K) which allows for about 790 mail items to be stored in core. Each additional 100K allows for about 750 additional mail items to be stored in core. For example, REG=1024 allows for about 3190 mail items to be stored in core.

**$TCP:POPPASSD namelist parameters**

The namelist parameters for $TCP:POPPASSD are MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, PORT, BUFSIZ, DROP,and TRACE. All of these parameters except PORT, BUFSIZ, DROP, and TRACE are the same parameters as define in $EML:MAIL.CONFIG for the Mail facility. The definitions for the exceptions are given below.

PORT=nnn Where nnn is the port number to run this server on. PORT=106 is the default. 106 is the standard port for EPASS.

BUFSIZ=n Where n is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=80.

DROP=n Where n is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value.

TRACE=t or f TRACE=t can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=f.

**Deleting mail from the server**

Some POP3 clients can be set up to leave mail on the server after it has been downloaded instead of deleting it from the server. The site may have a Mail policy that does not allow users to leave their mail on the server. The POP Mail Delete option in the Mail Profile facility, under General Mail Options can be set to Y to force mail to be deleted from the server after it is downloaded no matter what the client has requested. ADMIN 4 5 3 "Update site Mail Profile" can be used to set this option, and new mailboxes automatically get the set option. See the Mail Profile facility for further details on the POP Mail Delete option.

**Logging Features**

The namelist parameter LOG controls the logging features within the POP3 and POPPASS servers. If LOG=T, logging is done via the system log server BTRM. This logging information can be used to determine who is using these facilities, how much data is being transferred, and bad connection attempts. This data can help a site with mail management and capacity planning.

Both POP3 and POPPASS clients provide a userid and password to the respective server for the connection process. If the client enters an incorrect password for the userid, a log entry is written to the log server application name BADPW. This information can also show you who is trying to use the connection process to guess userids and passwords.

The POP3 server logs connection established and closed messages, sender verification failures for XTND XMIT support, and all other errors such as socket errors to the log server application name TCPIP. The connection closed entry also contains a byte count for all the mail downloaded to the client. A separate log message is written to the log server application names TCPIP and MAIL for each successful upload of mail for delivery as done by the XTND XMIT support.

**POP3 server XTND XMIT**

The POP3 server performs the XTND XMIT support by placing the incoming mail item into the reader queue, and letting RDMAILER deliver the item to the intended recipients. The reader queue information that RDMAILER uses to deliver the mail is hardcoded in the POP3 server. When RDMAILER reads the reader queue entry, it will produce a log server/console message like the following:

```
22:25 M405   7 *22.25.05 Mail received from POP3 at MUSIC to MUSIC
```

The POP3 server XTND XMIT support puts a Received: header in the uploaded mail item for traceability purposes. Also, if the POP3 client did not put a Date: header in the uploaded mail item, the MUSIC POP3 server will add a RFC822 Date: header.

**$TCP:POPPASSD and passwords in the clear**

The POPPASS server listens for incoming requests to change the password for a POP user. The client passes the required information (user name, old password, new password) to the server. After verifying the username and old password, the MUSIC POPPASS server runs a CODXXX5 job, the POPPASS helper program $TCP:POPPASSDH, to change the password.

This feature transmits unencrypted passwords over the network. The use of a dedicated port makes it easier for a network snooper to trap passwords off the wire. You may want to consider whether to implement the POPPASS server.

The commercial version of Eudora supports the POP3 XTND XMIT feature. For Eudora, defining the SMTP server address sends mail via SMTP. When this address is left blank, Eudora sends the mail via POP3 XTND XMIT.

### Delays and Timeouts

The POP3 and POPPASS servers have a hardcoded delay of one second to establish a connection. This is done so that people using the connection process to guess userids and passwords will not think a delay is a bad password feature. This slowdown in the connection attempt is a first level deterrent for breakins.

The POP3 and POPPASS servers have a hardcoded delay of three seconds when either server receives a bad password for the user connection. This slowdown in the connection attempt is a second level deterrent for breakins.

The default timeout value on socket operations both the POP3 and POPPASS servers see is controlled by the namelist parameter DROP. The default DROP=10 allows a 10 minute timeout before the connection is closed. DROP=10 is also the minimum timeout value allowed by RFC1939.

### Limiting How Often Users Can Connect to the POP3 Server

Often students configure their POP3 mail client to check their mail every minute. This is generally unnecessary and puts an added load on the POP3 server. Usually, mali can be checked every 10 minutes. The MUSIC/SP POP3 server POPFRQ namelist parameter can be used to define the frequency in minutes a user can access the POP3 server within a one day period. For example, if POPFRQ=10, a user is allowed access to their mailbox once every 10 minutes. If POPFRQ=0, users can access their mailboxes as often as they desire.

### Tracing Features

There are a number of features which can be used to help determine problems within the POP3 and POPPASS servers. Each server supports TRACE=T creating unit 6 output. The POP3 server also supports an alternate port assignment and it does MUSIC file I/O which can be trapped. All of these may be useful.

The namelist parameter TRACE when set to true will dump all read/write information from the socket and to any file to unit 6. Normally, unit 6 is not defined, thus output to unit 6 goes to the screen. The MUSIC /REC command may be useful in capturing this data. However, if output is lengthy, you may have to define a file to capture unit 6 output. This can be done by adding a /FILE 6 statement to the version of POPD you run. Example:

```
/FILE 6 N(*USR:POPD.TRACE) NEW(REPL) RECFM(VC) SP(500)
```

The namelist parameter PORT=110 sets the assigned port (110) for the POP3 server. While debugging the server, you could run it interactively. But you should set the PORT to a number different from 110, say 3110, so as not to interfere with the production version of your POP3 server.

The POP3 server also does MUSIC file system I/O in the XTND XMIT support. This type of I/O can be trapped by adding a /FILE statement to the version of POPD you run. Example:

```
/FILE MFTRACE N(POPD.MFTRACE) NEW(REPL) RECFM(VC) SP(500)
```

# The Phone Book Server (Ph)

The CCSO Nameserver (Ph) architecture is a client/server information model. The client sends commands to the server and the server sends responses to the client.

The Ph database is a telephone book containing local information about people or things. A client query would have the server return information stored in the database matching the query.

The Ph database is used to store a relatively small amount of information about a large number of people or things. Named fields are used in the database to associate a particular piece of information with an entry in the database.

The Ph specification was originally developed at the Computing and Communications Services Office (CCSO) at the University of Illinois at Urbana-Champaign. The Internet community has decided to publish a definition of the Ph specification to benefit the community. A number of servers and clients are available on various computing platforms.

Ph clients or a mail client containing Ph capabilities like the Eudora mail client provide easy access to local information such as email addresses or telephone numbers. Ph clients are available for a variety of platforms.

The Ph server is also be referred to as the Qi (Query Information) server.

## Ph Server

The MUSIC/SP Ph server is a TCP/IP server that allows a user using a Ph client or mail client containing Ph client capabilities to query the Ph database for local information. After fetching local email addresses from the database, these addresses can be used when sending mail. The Ph database offers a centralized place to store the local information for people and things.

## Ph RFCs

The MUSIC/SP Ph server complies with the Internet Draft draft-ietf-ids-ph-03.txt which can be found at ftp://ds.internic.net/internet-drafts/draft-ietf-ids-ph-03.txt

## Ph Database and its Management

The MUSIC/SP Ph database cannot be updated using Ph server commands. The MUSIC/SP program $EML:DIRECT.PUBLIC is used for this purpose.

The public nicknames file on MUSIC/SP, $EML:@NAMES, is used as the default Ph Database.

## Ph Clients

The Ph server has been tested with Eudora v3.0.1 for Windows (commercial software) and Microsoft's Windows 95 TCP/IP support.

## Latest Ph Documents

To see the Ph server documents coming from MUSIC/SP or for details of how to obtain the free Ph clients, just point your Internet web browser to our site by using the pointer:

```
http://MUSICM.McGill.CA/msi/http/ph.html
```

## Trademarks

Eudora is a registered trademark of the University of Illinois Board of Trustees, licensed to QUALCOMM Incorporated. QUALCOMM is a registered trademark and registered service mark of QUALCOMM Incorporated. All other trademarks and service marks are the property of their respective owners.

## Ph Server Installation

As the Ph specification evolves, updates to the Ph server are made available by anonymous FTP to MUSICM.McGill.CA. The following notes outline install procedures:

- If you have not installed the MUSIC TCP/IP support yet then you must do so first. Refer to the section "The Internet Super Server (INETD)" in this manual.

- Run MAIL.CONFIG to configure the Ph server program, $TCP:PH. See the Configuration Notes below for further details on the namelist parameters used by the Ph server.

- If you have logging on for TCPIP, you should add the following entries to the system log server BTRM define file, $PGM:SYSLG.DEFINE:

  ```
  TCPIP TO TCPIP1
  TCPIP1: FILE($TCP:TCPIP.) SP(100) SECSP(100%) MONTHLY SHARE
  ```

  You can turn off logging (LOG=F) if this is not important for your site. See the logging features in the Configuration Notes below for further details.

- If you want to limit full printable information to only the local community, create the file $TCP:@PH.LCLIP to define the local community. Copy the model file $TCP:PH.LCLIP.DUM to the file $TCP:@PH.LCLIP. Then edit the file $TCP:@PH.LCLIP and follow the instructions at the top of the file to define your local community. If this file does not exist, any query from any user not specifying return data will get all printable information for the matches to the query. If a local community is defined, any non-local queries not specifying return data will get just the alias, name, and email address fields returned.

- * Edit $TCP:INETD.PORTS and add a line at the bottom that reads:

  ```
  105  stream  10 $tcp:ph
  ```

  This definition uses the standard port 105 for the Ph server. This definition allows 10 clients at one time for the server.

- Edit $TCP:TCPIP.CONFIG and add port 105 to the end of the line starting with "RESTRICT_TELNET_PORTS". Ports specified on this line are separated by one blank. This stops MUSIC/SP users from telnetting directly to port 105.

- A change to the VM/SP TCP/IP configuration file PROFILE TCPIP may be required to allow access to port 105. Details of this change are beyond the scope of this document.

- Either re-ipl your MUSIC/SP system or issue a /RESET on the BTRM that runs INETD. (You will see this number on the console messages at startup time and on the console messages associated with each GOPHER or FTP connection.)

- If you want to make available to your users an updated list of the Ph servers around the world, add an AUTOSUB job to run $TCP:PH.SERVERS on a regular basis to build the information required for this query.

## Usage Notes

- The Ph phone book database is a centralized database for the site to store information about the local environment such as email addresses.

- The Ph database can be queried by users and the Ph server returns information matching the query to the user. For example, you are sending mail from your mail client on your workstation and you cannot remember the address of the intended recipient. You simply query the Ph server and have it return the address to you.

  Matching is not sensitive to upper or lower case letters and is normally done on a word by word basis. So, both the client query expression and the Ph database entry information are broken up into words, and the individual words are compared using exact matching. If the order of the words is important in a query, then the query string can be surrounded by "", whereby the complete query string is matched against information in the Ph database.

  The wild card character "*" within the query string can be used to represent zero or more characters. The wild card character "?" within the query string can be used to represent one or more characters.

  **Examples of query expressions:**

```
query name=Smith*  ;returns all matches with Smith* somewhere
                   ;in the name field
query alias=help   ;returns all matches with help somewhere in the
                   ;alias (i.e. nickname) field
Smith*             ;returns all matches with Smith* somewhere in
                   ;either the alias or name fields
"Donald Duck"      ;returns all matches to Donald Duck somewhere in
                   ;either the alias or name fields, where Donald
                   ;must immediately precede Duck in the field
```

- The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion.

- The MUSIC/SP Ph server limits the number of matches returned to a query. If this limit is exceeded by a query, the server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query.

- The MUSIC/SP Ph server can be defined to qualify who is a local user. The Ph server returns only the alias, name, and email address fields for each entry in the Ph database matching the non-local user's query if the query does not specify any return fields. The Ph server returns all information pertaining to the matches when a local user's query does not specify any return fields.

- The MUSIC/SP Ph server does not support the commands constituting the update features of the server. The following commands are supported:

```
status   - returns status of the database
siteinfo - returns info about the server site
fields   - returns the fields available in the database
id       - enters the given info in the log
set      - sets an option for this session
query    - requests a query with the given info
ph       - equivalent to the query command
quit     - disconnect from the Ph server
exit     - equivalent to the quit command
stop     - equivalent to the quit command
help     - returns help on the Ph server
```

- The MUSIC/SP Ph server "help" command has the following syntax:

  ```
  help [native] topic
  ```

  The word native is optional, and if it is not specified, the Ph topic is displayed. If that topic is not available, the "native" topic is displayed. If a topic is not specified, the Ph or native general help is displayed depending on whether native was given. The help text is stored on MUSIC/SP in files $TCP:PH.P* and $TCP:N*, where P and N represent Ph and native respectively, and "*" represents the topic name.

- Each field in the Ph database has an associated field descriptor defined in the file defined by the Ph server PHFLDS namelist parameter. A field descriptor includes the fieldname, the maximum length of the field, keywords describing the properties of the field, and a description of the field. The MUSIC/SP Ph server supports a subset of the keyword properties. See the topic "Adding Fields to the Ph Database" below for a list of these properties.

## Configuration Notes

**$TCP:PH Namelist Parameters**

The namelist parameters for $TCP:PH are MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, ERRFIL, PORT, BUFSIZ, DROP, TRACE, PHADMN, PHPSWD, PHFILE, HLIMIT, LCLIP, PHFLDS, and PHSRVS. The parameters MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, and ERRFIL are the same parameters as defined in $EML:MAIL.CONFIG for the Mail facility. The definitions for the exceptions are given below.

| | |
|---|---|
| PORT=nnn | Where *nnn* is the port number to run this server on. PORT=105 is the default. 105 is the standard port for Ph. If run interactively, the default is PORT=3105. |
| BUFSIZ=n | Where *n* is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=4096. |
| DROP=n | Where *n* is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value. |
| TRACE=T or F | TRACE=T can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=F. |

PHADMN=administrator_email_address

Where *administrator_email_address* is the email address of the system administrator. The Ph server command "siteinfo" returns information to the client about the server site. One of the fields returned is the system administrator's email address. The default is PHADMN='postmaster@MUSNOD', where MUSNOD is the MUSNOD namelist parameter.

PHPSWD=password_administrator_email_address
: Where *password_administrator_email_address* is the email address of the system password administrator. The Ph server command "siteinfo" returns information to the client about the server site. One of the fields returned is the system password administrator's email address. The default is PHPSWD='postmaster@MUSNOD', where MUSNOD is the MUSNOD namelist parameter.

PHFILE=phone_book_filename
: Where *phone_book_filename* is the filename of the database used for the phone book. The default is PHFILE='$EML:@NAMES'.

HLIMIT=n
: Where *n* is the maximum number of hits returned for each query. HLIMIT=25 is the default. HLIMIT=1 is the minimum value. HLIMIT=2147483647 is the maximum value. HLIMIT is set to 25 when HLIMIT is set to anything outside of this range. HLIMIT is meant as a partial deterrent from users trying to create a mail list by collecting all of the entries in the Ph database. HLIMIT=25 is the accepted value within the Internet community. See the topic Security Considerations below before changing this value.

LCLIP='filename'
: Where *filename* is the name of a list of IP addresses that represent the local domain. The default is LCLIP='$TCP:@PH.LCLIP'. The file contains one entry per line, and each entry can be a component of the full IP number (ie a network number). Blank lines or lines starting with a semicolon are considered comment lines. Any information on a line after a semicolon is considered a comment.

  Example:

```
132.206.120    ;defines all stations on 132.206.120 net
                as local
132.206.130.1  ;defines only 1 ip address on 130 net as
local
```

PHFLDS='filename'
: Where *filename* is a file whose contents defines the fields of the PH database. The default is PHFLDS='$TCP:PH.FIELDS'. Comments at the start of the file give instructions to add to file. Comment lines within the file start with a semi-colon in column 1. Do not remove the fields alias,email,name,phone,address,list. Add new fields following the list. When adding a new field, add 2 lines for each new field. First line format: field:max maximum_length keyword_properties Second line format: field:description_of_field There is no format checking so if the information is not typed in correctly, the Ph database may not be accessed correctly. The format can be verified with the PH client FIELDS command. The default type=person is used when the Ph database entry has not type. A new field name can be a maximum of 40 characters long. Each line can be up to 80 characters long.

PHSRVS='filename'
: Where filename is a file whose contents contains the Ph server response to a "query ns-servers" command as it would be responded to by the Ph server at ns.uiuc.edu (i.e. the mothership). The default is PHSRVS='$TCP:@PH.SERVERS'. Since the information contained in this file is not stored locally in the Ph database, a REXX exec, $TCP:PH.SERVERS, gets the data from the mothership and massages it as is needed so that it can be directly read and written to the client via the socket. This

exec could be scheduled as an AUTOSUB job to be run regularly.

*Note:* If you change this filename, you must edit the file $TCP:PH.SERVERS and change the variable file1 to agree with the filename entered here.

**Logging Features**

The namelist parameter LOG controls the logging features within the Ph server. If LOG=T, logging is done via the system log server BTRM. This logging information can be used to determine who is using these facilities, how much data is being transferred, and bad connection attempts. Since the phone book is typically used by mail clients to get local email addresses, the logging data can help a site with mail management and capacity planning. If LOG=F, logging is not done.

The Ph server logs established and disconnected connection messages, and all other errors such as socket errors to the log server application name TCPIP. The disconnected connection message also contains a byte count for all the bytes downloaded to the client.

**Mail Client Usage**

The Eudora mail client supports the use of the Ph server. Typically, the function is found in the Eudora client under the menus following Tools, Options, Directory Services or something similar. This feature brings up a window into which an input field for the Ph server and Ph request can be entered. Note that the Ph request field does not need to be prefixed the query or ph keyword, or with the field search name if the field is either "alias" or "name". For example:

```
Command => Smith
```

This query returns all entries in the phone book matching Smith either in the alias (i.e. nickname field) or anywhere in the names field.

The Eudora mail client supports the querying of the Ph server for a list of Ph servers. Typically, the "Server" button is found in the Eudora client under the menus following Tools, Options, Directory Services or something similar. The MUSIC/SP Ph server supports this feature.

**MUSIC/SP Ph Server Implementation Specifics**

The MUSIC/SP Ph server does not support the commands to update the Ph database. The MUSIC/SP program $EML:DIRECT.PUBLIC is used for this purpose.

Similarly, when the Ph server receives the "siteinfo" command from the client, the server returns the PHADMN and PHPSWD fields as set up in the MUSIC/SP configuration of the Ph server. These fields give the user contact information for the system administrator and system password administrator.

Fields within the Ph database are defined by the PH namelist parameter PHFLDS. The default is PHFLDS='$TCP:PH.FIELDS'. It is up to the site to get the extra fields into the Ph database and maintain these fields. See the description for the PHFLDS namelist parameter above under the topic "$TCP:PH namelist parameter" for further information. Also, the file $TCP:PH.FIELDS contains instructions on how to add fields to the Ph database. The topic "Adding Fields to the Ph Database" below provides further information. The keyword properties associated with each field are defined in the Ph specification as drafted by the IETF which can be found in file $TCP:PH.SPEC.

Only one field in the MUSIC/SP Ph database can possess the INDEXED keyword property. By default, the alias (i.e. nickname) field is given this property. It is up to the site administrator to keep the Ph database in INDEXED order (i.e. ascending, alphanumeric order). By default the Ph database is arranged such that the alias field is the first field in the database, so the command "SORT *" in $EML:DIRECT.PUBLIC can be used for this purpose. It is the site administrator's responsibility to maintain the Ph database in INDEXED

order for whatever field is given the INDEXED property.

The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion. See the topic Security Considerations below for information on this feature.

The MUSIC/SP Ph server limits the number of matches returned to a query. If this limit is exceeded by a query, the server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query. See the topic Security Considerations below for information on this feature.

The MUSIC/SP Ph server can be defined to qualify who is a local user. The Ph server returns only the alias, name, and email address fields for each entry in the Ph database matching the non-local user's query if the query does not specify any return fields. The Ph server returns all information pertaining to the matches when a local user's query does not specify any return fields. See the topic Ph Server Installation above for details on how to define the file $TCP:@PH.LCLIP. This file uses IP numbers to define local users. If this file exists, it is assumed that a local environment has been defined and any user not listed here is deemed a non-local user. If the file does not exist, all users are deemed local.

The MUSIC/SP Ph server supports a subset of the keyword properties. See the topic "Adding Fields to the Ph Database" below for a list of these properties.

The MUSIC/SP Ph server supports the querying for a list of Ph servers. This function is not supported through the Ph database as is done traditionally. Instead, the REXX exec $TCP:PH.SERVERS is used to get the data for the query from the main Ph server, ns.uiuc.edu, massage it, and store it in the file $TCP:@PH.SERVERS. When a client queries the MUSIC/SP Ph server for a list of Ph servers, the file $TCP:@PH.SERVERS is given to the client as the response. You may want to set up an AUTOSUB job to run $TCP:PH.SERVERS on a regular basis to keep the Ph servers list up to date. This feature is supported through the PHSRVS namelist parameter.

**Delays and Timeouts**

The default timeout value on socket operations for the Ph server is controlled by the namelist parameter DROP. The default DROP=10 allows a 10 minute timeout before the connection is closed.

**Security Considerations**

All data within the Ph database as defined on MUSIC/SP is public with respect to a Ph client. Privacy issues may need to be addressed from a local standpoint.

The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion. Using "*" in any query field is a convenient way for someone to build a mail list of the entire Ph database for the purposes of sending unsolicited mail. This feature makes it more work for a user to create a mail list from the Ph database.

The Ph database can be queried by any Internet user. Use of the file $TCP:@PH.LCLIP can define what information is returned on matches for non-local users. This file uses IP numbers to define local users. The topic Ph Server Installation above gives the details of how to install this feature. Essentially, with a local user, all data on a query match is returned to the user. With a non-local user, only the alias, name, and email address fields are returned to the query when the query does not specify any return fields. This feature provides limited information to non-local queries.

The MUSIC/SP Ph server namelist parameter HLIMIT is used to limit the number of hits returned to a query.

The default is HLIMIT=25. Although HLIMIT can be set to a value between 1 and 2147483647, HLIMIT=25 is the accepted value by other Ph servers on the Internet. This value is a balance between returning too few matches and returning the entire Ph database to a query. Setting HLIMIT to a high value means that any user creating a mail list from the Ph database would need fewer queries to create the list.

When the number of matches to a query exceeds HLIMIT, the MUSIC/SP Ph server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query.

## Adding Fields to the Ph Database

Fields within the Ph database are defined by the PH namelist parameter PHFLDS. The default is PHFLDS='$TCP:PH.FIELDS'. It is up to the site to get the extra fields into the Ph database and maintain these fields. See the description for the PHFLDS namelist parameter above under the topic "$TCP:PH namelist parameter" for further information. Also, the file $TCP:PH.FIELDS contains instructions on how to add fields to the Ph database. The keyword properties associated with each field are defined in the Ph specification as drafted by the IETF which can be found in file $TCP:PH.SPEC.

Each field in the Ph database has an associated field descriptor defined in the file defined by the Ph server PHFLDS namelist parameter. A field descriptor includes the fieldname, the maximum length of the field, keywords describing the properties of the field, and a description of the field. The MUSIC/SP Ph server supports the following subset of the keyword properties:

Always: Always printed in addition to whatever fields specified by the query

Any: Always searched by queries

Default: Printed if no return clause is given in the query

Indexed: Indexed field to allow searches on these fields especially efficient

LocalPub: May be viewed by anyone in the "local" domain space

Lookup: May be used in the selection part of a query

NoMeta: Wild card searches are disallowed

NoPeople: No entry of type "person" may include this field

Public: May be viewed by anyone

The Ph database is defined by the $TCP:PH namelist parameter PHFILE. The default is PHFILE='$EML:@NAMES'. The Ph database contents are tag based. The following is a sample Ph database entry:

```
:nick.ABC :emailid.prez@abc.com :name.ABC Company Inc
```

The fields are nick (i.e. alias), emailid, and name. These are default tags and must not be removed from the system. If a new field "fax" was desired, our sample entry might look like the following:

```
:nick.ABC :emailid.prez@abc.com :name.ABC Company Inc :fax.123-456-7890
```

After adding the "fax" field to those entries in the Ph database that required this field, the next step would be to define the "fax" field in the PHFLDS file. The following two lines would be added to the end of the PHFLDS file:

```
fax:max 40 Lookup Public Default
fax:Defines a fax number
```

Verify the changes by using the "fields" command of a Ph client to make sure the new "fax" field appears in the list of fields available. Now, the "fax" field is available for display or queries from Ph clients.

You can also test the changes by telnetting to Ph port, the default is 105, and typing "fields" to see the fields of the Ph database. Type "quit" to disconnect from the Ph server. You may be required to do the telnet from a system different from MUSIC if you configured the Ph server to not allow MUSIC users to telnet to port 105 as the installation notes suggest.

## Tracing Features

There are a number of features which can be used to help determine problems within the Ph server. The server supports TRACE=T creating unit 6 output. The Ph server also supports an alternate port assignment. All of these may be useful.

The namelist parameter TRACE when set to true will dump all read/write information from the socket to unit 6. Normally, unit 6 is not defined, thus unit 6 output goes to the screen. The MUSIC/SP /REC command may be useful in capturing this data. However, if output is lengthy, you may have to define a file to capture unit 6 output. This can be done by adding a /FILE 6 statement to the Ph server executor you run, normally $TCP:PH. Example:

```
/FILE 6 N(*USR:PH.TRACE) NEW(REPL) RECFM(VC) SP(500)
```

The namelist parameter PORT=105 sets the assigned port (105) for the Ph server. While debugging the server, you could run it interactively. But you should set the PORT to a number different from 105, say 3105, so as not to interfere with the production version of your Ph server.

# The Web Server (HTTPD)

This server handles HTML documents accessible through your Web browser. MUSIC's line-mode Web browser and information about how to create Web documents can be found in the *MUSIC/SP Internet Guide*.

## Tailoring and Changing Defaults

To change the default file matching or home page locations alter the following lines in $TCP:TCPIP.CONFIG:

```
HTTPD_ALLOWED_FILES *:*HTTP\*
HTTPD_ALLOW_PUBLIC no
HTTPD_ALLOWED_EXECS *:*HTTPEXEC\*
HTTPD_DEFAULT_PAGE $000:HTTP\HOME.HTML
```

HTTPD_ALLOWED_FILES — This entry defines the MASK for any file allowed to be accessed via HTTPD. For example, to allow the subdirectory HTTPD on userids that begin with dollar to be accessible, you specify $*:HTTPD* The default is *:*HTTP*

HTTPD_ALLOW_PUBLIC — This entry specifies whether public files not fitting the HTTP_ALLOWED_FILES mask, should be accessible. The default is no.

HTTPD_ALLOWED_EXECS This entry defines the MASK for any file allowed to be executed via HTTPD. This is specifically meant for forms support in html files. The default is *:*HTTPEXEC*

HTTPD_DEFAULT_PAGE This entry defines the default page to be transmitted when no page is specifically requested in the HTTPD connection. The default is $000:http\home.HTML

## Creating Alternate HTTP Servers

You can define alternate http servers on MUSIC. Typically the port number used for http service is 80. Alternate servers usually use numbers in the same range as the standard. For example a second http server may use the port number 81 etc.

The file $TCP:INETD.PORTS is where all servers are declared. The format is as follows:

```
80 stream 5 $tcp:httpd  <-- standard http daemon server
81 stream 5 your_own_file <-- your own http server
```

Refer to the section "The Internet Super Server (INETD)" in this manual for more details.

### Alternate HTTP Server Exec File

The following is description of how to set up an alternate http server exec file and how to specify values to tailor it.

Use $TCP:HTTPD as a template to create a file like the following:

```
/SYS NOPRINT,REGION=600,TIME=MAX
/LOAD XMON
httpD N($TCP:httpD.LMOD)
TRACE=F,PORT=N, .....
```

### Namelist Parameters

PORT=n Used to override the port number specified by httpSERVER in $tcp:tcpip.config.

TRACE=F/T Used to turn on tracing. This is used for debugging only. The output is directed to @tcpip.log and to @tcpip.buffers

MYSITE='site_name'
Used to override the httpSERVER in $tcp:tcpip.config.

DPAGE='default_page_name'
Used to override the http default page in $tcp:tcpip.config.
default='$000:HTTP\HOME.HTML'

LOG=T/F logs transactions.

HTFILE='mask' Mask is 1-64 character mask for allowed files. Used to override the default mask in $tcp:tcpip.config. Default *:*http*

BUFSIZ=N bufsiz is used to control the size of the buffering done to the socket. See comments in

$tcp:tcwrit.s for more details. Default is 2000.

TIMOUT=n timout value on reads and writes to socket, a value of 0 defaults to 30secs

PUBFIL=t/f allow access to files not matching the pattern (mask), but are otherwise public files or have the SHR file attribute.

**File Naming Conventions**

Only files that match the mask for allowed files will be transmitted.

Data (file) types currently supported are defined in the file $TCP:MIME.SUPPORT.

*Notes:*

1. EBCDIC refers to files that are created and maintained on MUSIC or are in an ebcdic format.
2. ASCII refers to files that have been created in ascii and imported to MUSIC. These files can not be modified using the MUSIC EDITOR. They can be read using VIEW and the command "ascii" to provide a translated view of the file.
3. binary files are similarly imported to MUSIC but are not in a readable format.

**Convention for URL's:**

Rule 1

```
http://musicdns/userid|file/file/file
```

will be treated as

```
http://musicdns/userid:file/file/file
```

Rule 2

```
http://musicdns/~userid/file/file/file
```

will be treated as

```
http://musicdns/userid:file/file/file
```

## Changing, Adding and Deleting MIME Types

The Software is sent out with as many known MIME and file types at packaging time. It may, however, become necessary to make changes to the list of file types. The list used by the HTTPD server is stored in the file $tcp:mime.support, and looks as follows.

```
;for type enter either bin or text
;
;type  MIME TYPES                SUFFIXES
;----  --------------------      -----------------
text   text/plain                .txt,.text,.etext
bin    text/plain                .atext, .atxt
```

```
text   text/html                 .html, .htm ITS
bin    text/html                 .ahtml, .ahtm
bin    image/gif                 .gif
bin    image/jpeg                .jpg,.jpe,.jpeg
;
bin    audio/wav                 .wav,.wave
bin    audio/au                  .au
bin    audio/x-midi              .mid
bin    image/x-tiff              .tif,.tiff
bin    video/mpeg                .mpg,.mpe,.mpeg
bin    video/quicktime           .mov
bin    video/msvideo             .avi
bin    application/postscript    .ps,.eps,.ai
bin    application/x-rtf         .wri
bin    application/octet-stream  EXE bin class
```

To make changes to the list, edit $tcp:mime.support and make the required change, deletion or addition. The syntax is

```
type  mime_type  suffixes
```

*Notes:*

1. For "type" only bin and text are allowed. This simply flags that the file is either EBCDIC (readable on MUSIC) or binary (not readable on MUSIC). When "bin" is chosen the file contents are sent to the client with no filtering or translations. While "text" indicates that the file contents are to be both translated to ascii and that a CRLF (carriage return/line feed) are to be appended to each line of the file as it is transmitted.

2. "mime_type" should be in lower case.

3. "suffixes" do not need commas or dots. Care should be taken that no standard suffixes are removed or corrupted. This could result in incorrect transmission to the client. Of special note, is the "ITS" suffix. This suffix is internal to MUSIC and is used in naming Indexed-Text-Searchable files.

## The General Flow of HTML Forms Document Programs

1. Call to GTFORM to fetch the response string from the client. The string will have a series of name/value pairs seperated by ampersands (&). The following is a sample from the example presented later in this section:

   ```
   last name=&first name=&department=&email=<phone>=&
   month=April%2095&day=Monday&time=10%3A00-12%3A00%20am
   ```

   Note that special characters are transmitted as %xx, where xx is a hexidecimal number representing an ASCII character. The RDFOM and RDNUM routines automatically convert these for you.

2. Calls to RDINFO and either RDFORM or RDNUM will retrieve the name/value pairs. Please see the REXX sample program to help you understand the technique.

3. The program performs what ever process is necessary.

4. Calls to ADLINE are used either during or after processing to construct a response to the user.

5. Call to RSPOND is used to signal to the HTTPD server that processing is complete and that a response

is available for transmittal to the client.

## Subroutines for Processing HTML Forms

Programs written to handle html forms will use some or all of the following routines (details of each routine can be found below). These routines can be used in any programming language including REXX.

| | |
|---|---|
| GTFORM | get the client's returned string of variable names/ values from what the user filled in. |
| RDFORM | given a variable name return its value. |
| RDNUM | get the name/value strings of the ith pair. |
| RDINFO | tells you how many name/value pairs there are, as well as the maximum lengths of the name strings and value strings. |
| RDPOS | returns the positions and lengths of the name/value pairs. |
| ADLINE | add a line to the response to the client |
| RSPOND | used to tell HTTPD server that program has completed. The text passed back by adline will be transmitted to the client as a response to the user. |
| RDRSET | resets all pointers, variables etc. |

## Descriptions of HTML Subroutines

### Parameter Descriptions

The following parameters are used in the subroutines below.

| | |
|---|---|
| RC | returns the length of prmstr, if KWORD is not found, -1 is returned as the value. |
| KWORD | A 1 to 32 character keyword to be searched for |
| KLEN | length of KWORD |
| STRING | The string from the client with the forms response. |
| STRLEN | The length of string. |
| PRMSTR | The character string associated with KWORD. |
| PRMLEN | The maximum length of PRMSTR. |
| VALUE | Integer value of PRMSTR (where applicable) |
| VARNUM | ITH variable in the form |
| TOTVAR | total number of variable/value pairs in the form. |
| MXVARL | the length of the longest variable name. |

MXVALL the length of the longest value string.

VARPOS position of the ith variable.

VARLN length of the ith variable

VALPOS position of the ith value string

VALLN length of the ith value string

**GTFORM**

This subroutine reads from the file a string returned from the client with the variable/value pairs. Calling sequence:

```
CALL GTFORM(RC,STR1,STRL1,STR2,STRL2)
```

Arguments:

RC mfio return code in case of error.

STR1 1-32760 text string to be retrieved from the file (line 1). this is in the form keyword1=text1&keyword2=text2

STRL1 MAXIMUM length of STR1

STR2 1-256 text string to be retrieved from the file (line 2) this is in no particular form.

STRL2 MAXIMUM length of STR2

**RDFORM**

This subroutine is used to get the value of a known variable name. Given KWORD, PRMSTR is returned as the value of KWORD. Calling Sequence:

```
CALL RDFORM(RC,STR1,STRL1,KWORD,KLEN,PRMSTR,PRMLEN,VALUE)
```

Some special characters are encoded with a preceding % (per cent). For example an astersik (*) will come to us as an encoded %2A, where 2A is the ascii hex value. We will convert "%2A" to "*".

Example:

STRING contains the following, and is of length 64
password=secret&fromemail=me@here&title=Duh..%20title&message=Text

```
CALL RDFORM(RC,'password',8,STRING,80,PRMSTR,80,VALUE)

returns: RC     6       (length of prmstr)
         PRMSTR secret  (character string associated with password)
         VALUE  0       ('secret' is not an integer)
```

Note that the keyword must be in the same case as was used in the HTML document form.

**RDNUM**

This subroutine is used to get the variable name and its value from the string. VARNUM is used to point to the ith pair. Calling sequence:

```
CALL RDNUM(RC,STR1,STRL1,KWORD,KLEN,PRMSTR,PRMLEN,VALUE,VARNUM)
```

**RDINFO**

This subroutine is used to return the number of variable/value pairs and the length of the longest variable name and the longest value string. Calling sequence:

```
CALL RDINFO(RC,STR1,STRL1,TOTVAR,MXVARL,MXVALL)
```

You must call one of RDINIT, RDFORM, or RDNUM before you call RDINFO.

**RDPOS**

This subroutine is used to return the character positions of the variable/ value pair pointed to by VARNUM. Calling sequence:

```
CALL RDPOS(RC,STR1,STRL1,VARNUM,VARPOS,VARLN,VALPOS,VALLN)
```

**ADLINE**

Adds line of text to a file. If the file is not open the file will be opened fro the caller. Calling sequence:

```
CALL ADLINE(RC,TEXT,TXTLEN)
```

Arguments:

RC mfio return code in case of error.

TEXT 1-256 CHARACTER text string to be written to the file.

TXTLEN length of TEXT.

**RSPOND**

This routine is used to signal the HTTPD server that processing is complete and take the text lines provided by ADLINE, and respond to the user. Calling sequence:

```
CALL RSPOND(RC,GEN)
```

Arguments:

GEN =0 means that we pass the contents added via ADLINE routine back to HTTPD
=999 means that we will use a generic "Everything is ok" file.
=888 means that we will use a generic "Something is wrong" file.

**RDRSET**

This subroutine is used to zero all counters and pointers into string. Calling sequence:

```
CALL RDRSET
```

# The Gopher Server (GOPHERD)

GOPHERD is the name of MUSIC/SP's Gopher server. A gopher server provides access to documents residing on a host or server to a gopher client running on another computer. The gopher client may be running on a wide variety of platforms and operating systems. This section discusses how to set up documents on MUSIC/SP for access by gopher clients. (MUSIC's Gopher client is documented in the *MUSIC/SP Internet Guide*.) To know more about gopher in general please refer to the document, "The Internet Gopher Protocol" (RFC 1436).

## Data and Documents on Gopher

Gopher supports several types of data or documents. These item-types are defined by the following characters:

0 Item is a file
1 Item is a directory
2 Item is a CSO phone-book server
3 Error
4 Item is a BinHexed Macintosh file.
5 Item is DOS binary archive of some sort. Client must read until the TCP connection closes. Beware.
6 Item is a UNIX uuencoded file.
7 Item is an Index-Search server.
8 Item points to a text-based telnet session.
9 Item is a binary file! Client must read until the TCP connection closes. Beware.
+ Item is a redundant server
T Item points to a text-based tn3270 session.
g Item is a GIF format graphics file.
I Item is some kind of image file. Client decides how to display.

The item-types currently supported on MUSIC are 0, 1, 7, 8, 9, T, g, and I.

## Gopher Directories (menus)

A gopher server tells the a gopher client what it has available through directories transmitted to the client. Directory items transmitted to the client look as follows:

```
item-type caption|selector|site_name|port_number
```

where:

item-type defines the item (directory, document, etc.

caption is the descriptive text that the client displays to assist the user in making his selection.

selector this is a string that the client sends to the server in order to select the item.

| | |
|---|---|
| site_name | internet address name (can be an IP address) This is the site to connect to to get this item. |
| port_number | This is the port number to be used to establish the connection with site_name. |
| \| | This character represents the tab character hexadecimal '05'. MUSIC's GOPHERD automatically converts the '\|' (vertical bar) to hex 05. '\|' is used as a delimiter between elements of the the directory definitions. You can substitute tilda (/ xA1), exclamation mark (! x5A), not sign (• x 5F), or cent sign ( x4A) for '\|'. |

The following is a sample gopher directory (menu) from RFC1436.

```
0About internet Gopher|Stuff:About us|rawBits.micro.umn.edu|70
1Around University of Minnesota|Z,5692,AUM|underdog.micro.umn.edu|70
1Microcomputer News & Prices|Prices/|pserver.bookstore.umn.edu|70
1Courses, Schedules, Calendars||events.ais.umn.edu|9120
1Student-Staff Directories||uinfo.ais.umn.edu|70
1Departmental Publications|Stuff:DP:|rawBits.micro.umn.edu|70
```

The client will display this as follows:

```
About Internet Gopher
Around the University of Minnesota...
Microcomputer News & Prices...
Courses, Schedules, Calendars...
Student-Staff Directories...
Departmental Publications...
```

The menu above will be displayed differently by different clients. For example some clients may number the items, while others may use icons and allow mouse support etc. In our display above '...' represents those items that are directories. Note that when the item refers to the home directory of a gopher server or gateway, there is no selector string, hence the adjacent '|' in the examples above.

## Home Gopher Menu for GOPHERD

To provide a home directory (menu) for your MUSIC/SP gopher server, create or update the file "$tcp:gopherd.menu". What you place here is what gopher clients will see when they gopher to your site. Just follow the example above or consult RFC 1436 to make your own home directory.

## MUSIC/SP Selector String Conventions

Gopher server/clients operate in a mode, that is sometimes referred to as "stateless and connectionless". This simply means that the client (user) is not connected to the gopher host as one might be connect via FTP or Telnet. Rather, after each request the connection between the server and client is severed. For every request a new connection is made to the site named on the item. After connecting to the gopher server the client passes it either a blank line to indicate that it wants its home directory or the selector string extracted from the directory item.

The selector string can be used by the gopher server to establish what the data is and where it is located. Although the client has been told what item-type to expect, the item- type is not returned to the server by the client. This means that the server has to know from the selector string or via some other method what the data type is.

MUSIC/SP uses several flags in the selector string for identifying the kind of request being made.

-BBS indicates that the item is a BBS text menu file
-BB1 indicates that the item is a BBS itemized menu file
-BIN item is a binary file
-GOP indicates that the item is a standard gopher directory
-GIF item is a GIF file
-IMG item is an image file
-ITS item is an INDEX TEXT SEARCH facility
-IT1 item is the pointer to the result of a text search
-PGM item is an executable MUSIC file (not supported)
-TXT item is a straight text file (can be a BBS file)

**Examples:**

```
7Telephone Book|-ITS/tb/|site_name|70
```

defines a gopher Indexed Text Search item called "Telephone Book" and the executor file name is TB. This is all that is required to define an ITS application for the gopher server.

```
1infoMcGill|-BBS/CW99:@inf.main.menu|site_name|70
```

defines a gopher directory item called "infoMcGill", which is a MUSIC CWIS application. Once the item is selected by the client, all other gopher menus required to display the text and CWIS menus in infoMcGill will be automatically generated by MUSIC's gopher server.

```
gCampus Map|-GIF/$ABC:MAP1|site_name|70
```

defines a gopher item-type of GIF in the music file "$ABC:MAP1".

```
1More documents|-GOP/$ABC:GOPHER.MENU2|site_name|70
```

defines a gopher item-type of directory (gopher menu) in the music file "$ABC:GOPHER.MENU2".

*Note:* '-BB1', '-IT1', and '-PGM' are used internally only.

## Gopher Support for MUSIC/SP BBS Facilities

As was stated earlier, to point to a BBS for gopher clients you simply point to the main menu of the BBS. For example McGill's BBS (Campus-Wide Information System) called infoMcGill can be defined as follows:

```
1infoMcGill|-BBS/CW99:@inf.main.menu|musica.mcgill.ca|70
```

where:

| | |
|---|---|
| 1 | item-type directory (menu) |
| infoMcGill | is the caption seen by the user of the gopher client |
| -BBS/ | tells MUSIC's gopher server to treat the item as a BBS. Remember that this string is sent to the server by the client in order to request the item. |
| cw99:@inf.main.menu | is the name of the MUSIC file where the main menu for infoMcGill resides. |
| musica.mcgill.ca | address of McGill Universities MUSIC system. |

70 port number

This is all that is required to establish access to the entire BBS. From this point on the directories and item-types will be automatically generated by the MUSIC gopher server. This includes automatically linking to other BBS and ITS applications if they occur as topics in the BBS.

## Making Files Accessible via Gopher

The Systems Administrator can define a MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessable you specify $*:gopher*

The default is "$pub:*". The following is an example of the parameter that is placed in the $TCP:TCPIP.CONFIG file.

```
GOPHERD_ALLOWED_FILES $PUB:*
```

## Special File Flags for BBS Support

The gopher server uses several flags and clues in the content of the BBS files. Most of these flags are already used in the normal definition of BBS files. These are:

)menu the file is an itemized menu file

)Gmenu the file is a text menu file. These are menus that have highlighted text pointing to BBS topic files (see notes below).

)Gopher the file is a gopher directory (not often used) If "-GOP" flag is not used in the item definition example:

```
1More documents|-GOP/$ABC:GOPHER.MENU2|site_name|70
```

then you must place ")gopher" as the first line of the file.

)TEXTFILE the file is a straight text file. As with )gopher you can either specify the item as a text file via the "-TXT" flag or you must place ")textfile" in the document to correctly identify the item-type.

)INETACC on/off
This statement enables or disables access to particular topic files when GOPHER is being used. The default is "on".

**Notes:**

Most BBS facilities do not have )gmenu specified in the text menu files as this is not required to run BBS as an executable program on MUSIC. However, it is required when the BBS is accessed via gopher. You should place a )gmenu line in all text menu files of a BBS before making it available on gopher.

A text menu file is a BBS topic file with one or more columns of topic names and a companion descriptive text.

Example:

```
Place the cursor on the topic of your choice and press enter

+?library hours -? schedule for campus libraries
+?library loans -? how to take out a library book
+?library tours -? tours of the library facilities
```

should be modified as follows:

```
)gmenu
Place the cursor on the topic of your choice and press enter

+?library hours -? schedule for campus libraries
+?library loans -? how to take out a library book
+?library tours -? tours of the library facilities
```

)gmenu does not affect the BBS's operation as an executable program on MUSIC but is used as an aid in auto detecting the file types.

A simple method to retro-fitting )gmenu into the files that are text menus is to use FLIB to bring up a list of topic files for a given BBS. For example, infoMcGill's topic files (see above) can be accessed by typing the following "flib cw99:@inf.*".

## Tailoring and Creating Alternate Gopher Servers

You can define alternate gopher servers on MUSIC. Typically the port number used for gopher service is 70. Alternate servers usually use numbers in the same range as the standard. For example, a second gopher server may use the port number 71 etc.

The file $TCP:INETD.PORTS is where all servers are declared. The format is as follows:

```
70 stream 5 $tcp:gopherd       <-- standard gopher daemon server
71 stream 5 your_own_file      <-- your own gopher server
```

Refer to the under the topic "The Internet Super Server (INETD)" in this manual for more details.

The following is a description of how to set up an alternate gopher server exec file and how to specify values to tailor it.

### Usage

Create a file like the following, you can use $TCP:GOPHERD as a template.

```
/SYS NOPRINT,REGION=600,TIME=MAX
/LOAD XMON
GOPHERD N($TCP:GOPHERD.LMOD)
TRACE=F,FMENU='FIRST_MENU',PORT=N, .....
```

**Namelist Parameters**

PORT=n Used to override the port number specified by GOPHERSERVER in $TCP:TCPIP.CONFIG.

TRACE=F/T Used to turn on tracing. This is used for debugging only. The output is directed to @TCPIP.LOG and to @TCPIP.BUFFERS

MYSITE='site_name'
Used to override the GOPHERSERVER in $TCP:TCPIP.CONFIG.

LOG=T/F logs transactions to $TCP:TCPIP.LOG.

FMENU='menu_name'
Used to set a first menu other than $TCP:GOPHERD.MENU. It can be the main menu of a BBS or the standard gopher syntax. For example, to make the help facility the first menu, you would specify FMENU='$hlp:@go.main.menu'.

ALLOWF='file_pattern'
is used to override GOPHERD_ALLOWED_FILES from $TCP:TCPIP.CONFIG.

This entry defines the MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessible you specify "$*:gopher*".

The default is "$pub:*".

BBSLOG='userid/prefix'
this parameter can be used to produce a bbstat like log of usage. The string must be in the form 'userid:@xxx.' This makes GOPHERD logging compatible with BBS logging.

# The Finger Server (FINGERD)

This handles the "finger" requests that enable users on remote hosts to find out who is signed on to your MUSIC system. It basically sends them the output of the WHOSON utility. If you wish this service to be available, configure INETD.PORTS to run $TCP:FINGERD on stream port 79.

```
79    stream   2  $tcp:fingerd
```

# IRC Client

The IRC client must be configured before using it. To configure the IRC client, simply edit the file $TCP:IRC and insert the names of your servers as: server='server-name' on multiple lines.

Each server will be tried in succession until a server accepts a link with your client. Thus, you should place your servers in order of preference. Note that the IRC server site must usually agree to accept communications from your site's clients.

# Writing Your Own Servers

It is possible to write your own servers using the socket interface. For more information about the socket interface and writing socket applications see the file $TCP:SOCKETS.DOC. There are also two scaled down samples distributed with the system that can be used as models. See the files $TCP:DEMO.SERV.FORT and $TCP:DEMO.SERV.REX for details.

Production servers usually run as BTRMS scheduled by INETD and it is very difficult to debug them in this mode. For this reason the GETCON routine is provided to allow you to run the server independently of INETD on a regular terminal session so you have all the system debugging tools available. During testing use the GETCON routine to define the port:

```
S=GETCON(port)
```

This will initialize the specified port and wait for a connection to be made. The socket number (S) is set. This routine allows only one connection at a time on a given port, but this is usually sufficient for testing purposes.

When you want to put the server into production replace GETCON by GETSOC:

```
S=GETSOC()
```

This routine gets the socket number from INETD. Define the port that the server will use in the $TCP:INETD.PORTS file and restart INETD to activate the new server.

If you wish to code socket application in C, you must include a call to CARGENV (no parameters required) in order to cause the socket routines to automatically convert the arguments passed to assembler calling conventions. In this case, use the normal C calling conventions as defined by the SOCKET.HDR header file.

# News Reader

The news reader (RN) allows users to access information on a remote network news feed. The program must be configured prior to use. RN establishes a TCP connection with the news feed machine and uses the NNTP protocol to access the information. News items are fetched over the network as required. News submissions can be made by direct posting to the server, if that is allowed, or through mail.

The RN program is configured by editing the file $TCP:TCPIP.CONFIG and setting the following parameters.

NEWSFEED The internet address or name of a host providing an NNTP news feed.

NEWSGROUPFILE The name of a file to contain the list of NNTP news groups.

NEWSDROP_0_POSTS
Specifies whether news groups with no postings are dropped from the news groups list. (yes/no)

NEWSGROUPNAMESIZE
Maximum length of news group name before truncation.

ORGANIZATION The name of your organization.

TIMEZONE Your time zone. (GMT, EST, MST, etc.)

## News Groups File

This file is simply a sequential list of the news groups that are available. The $TCP:NEWSGROUP program contacts the NNTP server and builds this file based on information from the server. This process should be automated to update the list periodically (once a day perhaps). The AUTOSUB utility can be used to provide this automation by submitting $TCP:NEWSGROUP to batch when appropriate. The SYSCOM and FILES privileges are required.

*Note:* ADMIN (4 9) can be used to add $TCP:NEWSGROUP to the automatic maintenance jobs that AUTOSUB processes.

## $TCP:NNTP.SERVICE.S

These routines provide the TCP/IP transport services for RN and the NEWSLIST program. There are several return codes that it generates. These are:

-101 - bad server name
-102 - could not determine local site IP address (problem with TCP/IP)
-103 - could not determine remote site IP address
-999 - connection closed by remote host

# TCPSTAT - TCP Applications Analysis Facility

TCPSTAT is used to perform basic usage analysis of TCP/IP applications and facilities.

TCPSTAT can be used to either provide a display of a daily TCP/IP statistics log or to produce a statistical report on usage.

Select one of the following by typing its number in the command area.

```
1. Browse a TCP daily statistic file.
2. Produce a usage and activity report for TCP applications you want
```

After you have made your selection, press ENTER. The following sections describes the screens that assist you in customizing the browse or the report you want to generate on the usage patterns for a particular TCP/IP application.

## Browsing TCP Statistics Logs

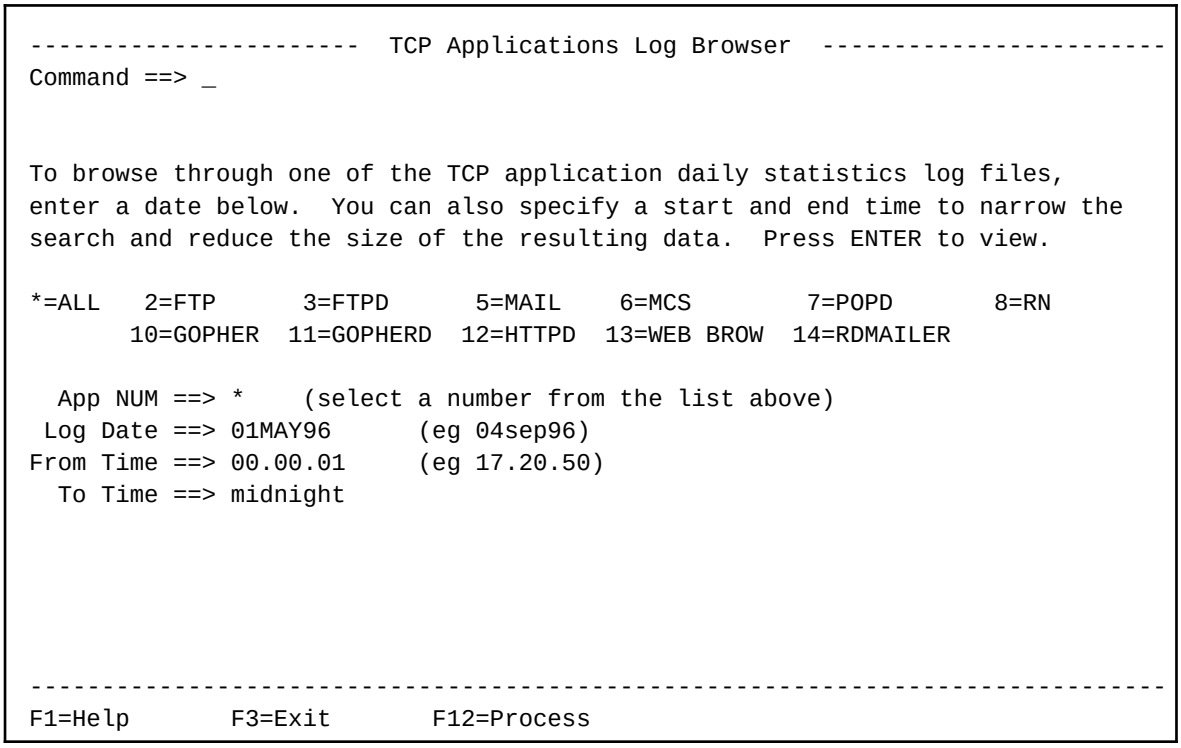


*Figure 16.1 - Browsing TCP Statistics Logs*

App NUM — You can choose to browse through one of the TCP application daily statistics logs by selecting one of the numbers below that correspond to TCP applications. "*" can be used to provide a report for all applications.

```
   *=ALL   2=FTP      3=FTPD      5=MAIL    6=MCS        7=POPD
  8=RN
            10=GOPHER  11=GOPHERD  12=HTTPD  13=WEB BROW  14=RDMAILER
```

Log Date — The daily log to be browsed is selected by entering a date on the "Log Date" field in the form ddmmmyy (28APR96).

From Time
To Time — You can also specify a start and end time to narrow the search and reduce the size of the resulting data.

When you have filled in the fields with the appropriate values press ENTER to browse the log file.

## TCP Applications Statistics Report Generator

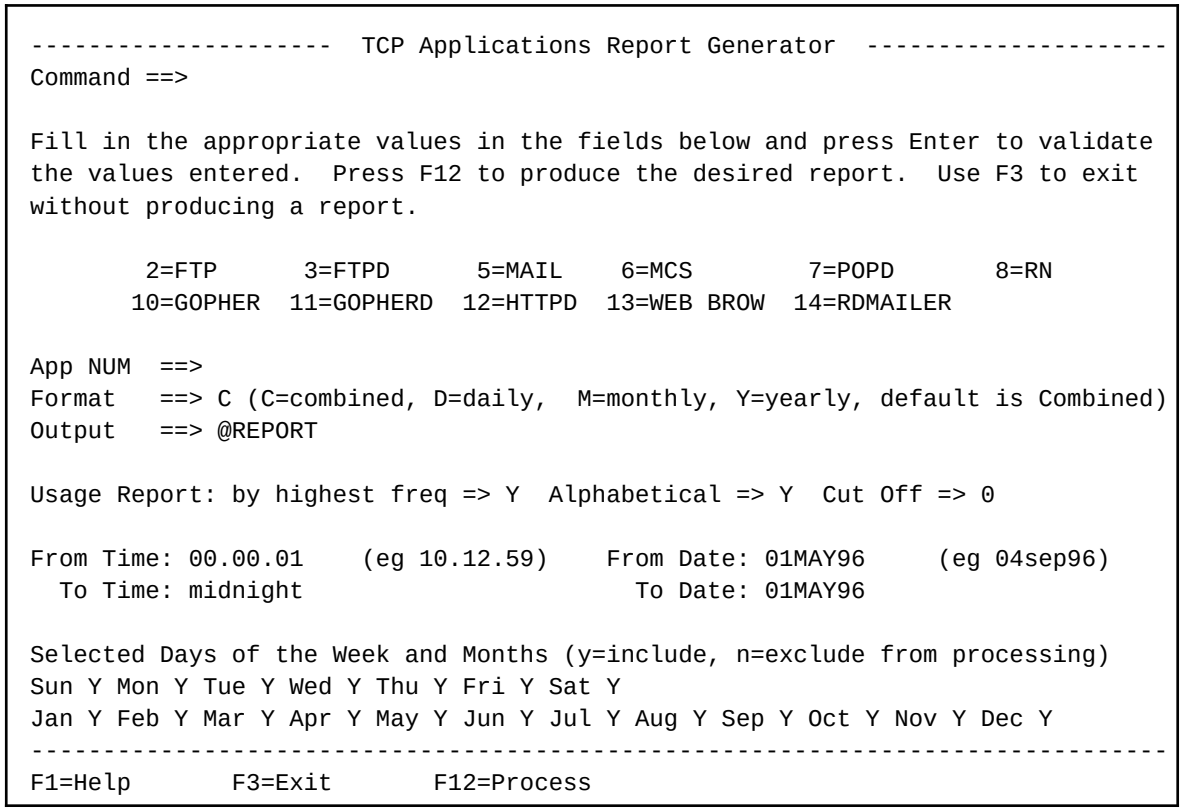


*Figure 16.2 - TCP Applications Statistics Report Generator*

You can obtain a statistical report on one of the TCP/IP applications. The report can be based on daily, monthly, yearly or a combined format. The report is divided into four parts. The first part reports to you all the values and parameters you entered to generate the report. The second part provides a frequency and usage chart based on the hour of the day. Column summaries are provided at the bottom of the chart. The third part provides summaries and averages for the entire report regardless of the report type. The fourth part provides a usage report of resources particular to the application.

App NUM — To select a statistical report for one of the TCP/IP applications enter its number.

Format — You can choose on of four formats for the report: daily, monthly, yearly and combined. The frequency charts based on the hour of the day will be produced based on the report format. The other parts of the report will be cumulative values based on all the data analyzed.

Output — You can specify a MUSIC file name to which the report is to be written. The default is "@report".

Usage Report — Each TCP/IP application has some resource or usage item that can be reported. For example HTTPD (the WEB server) downloads files. A frequency report on files downloaded is provided in order of high frequency of usage and by alphabetical order. Most applications will also report the IP addresses in the same fashion. This will allow you to determine what resources your system serves, what your users are fetching from the network and what IP addresses are involved.

You can also cancel or limit this feature of the report by specifying; "Y" yes produce usage report or "N" no do not produce the usage report. You may also want to reduce the output generated by limiting reporting to only those items that occur at least x number of times.

Use the "Cut OFF" field for this.

From TIme
To time — You can limit the report to a specific date and time range.

Selected Days... — It may be desirable to include or exclude specific months or week days. This section allows you to do this.

# Making MUSIC look like an Internet Host

## Introduction

It is possible to make the MUSIC virtual machine look like a physical internet host that functions independently from the VM system on which it is running. Currently the main advantage of this is that both MUSIC and VM can run servers on the same ports. For example, VM could run an FTP server on port 21 giving users access to the CMS file system and at the same time MUSIC could run an FTP server on port 21 giving access to the MUSIC file system.

To do this you must run two copies of TCP/IP under VM connected via an IUCV link. One copy of TCP/IP is used by the CMS users and the other is for use by the MUSIC virtual machine. This is similar in concept to using two RSCS virtual machines for BITNET connectivity, but has the significant advantage that you do not have to modify TCP/IP. The following diagram, figure 16.3, illustrates the situation.

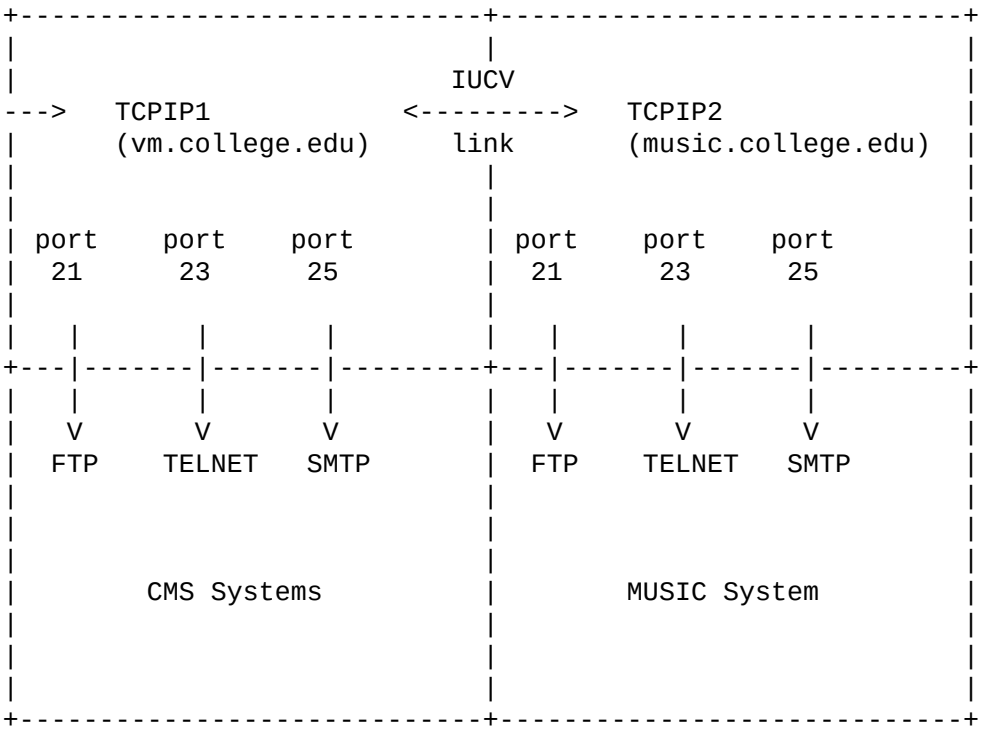


*Figure 16.3 - Setting up TCP/IP for MUSIC*

## Setting Up an IUCV Link

The following assumes that you have already installed a TCP/IP system under VM for your CMS users.

If your site must provide TCP/IP services to your CMS users and your MUSIC users from their respective systems, one of the choices in the TCP/IP configuration for this setup is to run two TCP/IP systems. These two systems would run on separate virtual machines under the same VM, using an IUCV (point to point) link. One system is made available for access to and from CMS and the other system for access to and from the MUSIC system.

This section describes how to set up an IUCV link between two virtual machines running TCP/IP under the same VM.

**Requirements**

This can all be done using the IUCV driver included in the VM TCP/IP (alias FAL) software and does not require a hardware interface. You will need to get a new, unused subnet or network number (if your site isn't subnetted) assigned to the system you are going to create. You cannot use the same subnet(network) as the TCP/IP system you have already installed since the Internet Protocol specifies that different "physical" network types must have different subnet(network) numbers.

This support requires at least the 9201 service for the MUSIC/SP 2.3 system and the IBM VM TCP/IP V2, program number 5735-FAL. It is suggested that you have an installed TCP/IP system running under VM before proceeding with this procedure.

Along with the directions described below, you may also wish to consult the IBM TCP/IP Version 2 for VM: Planning and Customization manual, SC31-6082, for further details.

**Routing Notes**

If your campus is using STATIC ROUTING (ie a gateway/router has a file containing a list of the networks available for your site that it uses when it boots), then ignore the comments about the ROUTED virtual machine. You do not require this userid. However, you will need to have a gateway/router on your campus announce a static route to the new subnet(network) via the installed TCP/IP system's physical interface. Alternatively, you could add a static route in all of the other hosts that require access to the new host. In the configuration step below, this setup is referred to static routing.

If your campus is using DYNAMIC ROUTING (ie a router(s) discovers the networks available for your site via a routing protocol) with the Routing Information Protocol (RIP), then you will have to run ROUTED on both ends of the IUCV link. ROUTED can be used to manage the IUCV link only if both ends of the link run ROUTED. If only the installed TCP/IP system end runs ROUTED, it will not receive routing updates from the other end and it will assume the link is down. ROUTED will then delete all of the routes in its tables for the IUCV link. This problem is inherent to the RIP protocol and cannot be prevented. Therefore, you must run ROUTED on both ends of the IUCV link. In the configuration step below, this setup is referred to as dynamic routing.

Please consult the chapter on "Configuring the ROUTED virtual machine" in the TCP/IP VM Planning and Customization manual for further details.

**TCP/IP V2R0 support notes**

These notes reflect the fact that we have TCP/IP V2R2 installed. TCP/IP V2R2 has some notable changes over V2R0, one of which is important with respect to these notes on the implentation of an IUCV link.

V2R2 has both a 591 minidisk which contains the execs and modules required by TCP/IP servers, and a 592 minidisk which contains the files required by TCP/IP clients. V2R0 has only a 592 minidisk which contains all of the files required by TCP/IP servers and clients.

If you have V2R0 installed and are planning to implement the IUCV link as documented here, just ignore the references to the 591 minidisk. It is strongly advised that your site upgrade to V2R2, which contains the numerous bug fixes to V2R0.

### A Sample Setup to use as a Model

We use our setup at the MUSIC Product Group as a model. The following diagram describes our setup. The addresses and domain names are used throughout this document in examples.

McGill University was assigned the domain name McGill.Ca. The MUSIC Product Group was given the subdomain MPG.McGill.Ca. The 132.206.120 network is our ethernet link to the campus router which connects us to the Internet. The 132.206.131 network is our point-to-point (IUCV) link connecting our two TCP/IP systems running under the same VM system.

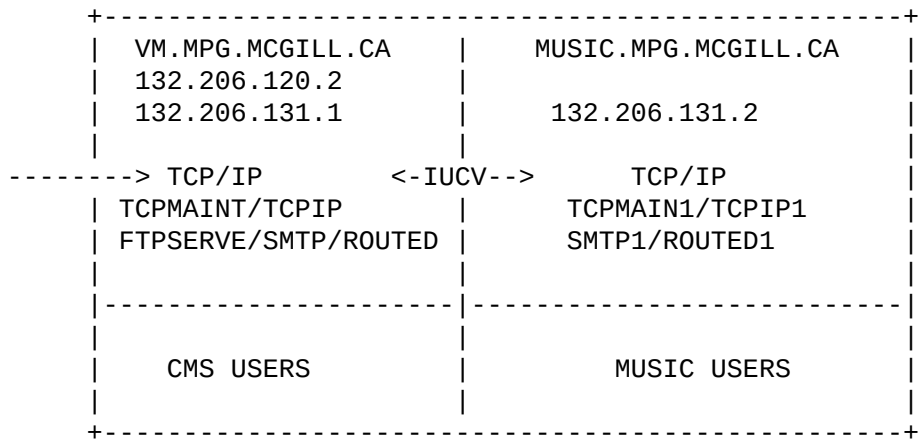


*Figure 16.4 - Sample Setup for TCP/IP*

## IUCV Steps

There are five steps for this support:

1. Add the new system's domain name into the campus name server.
2. Define the set of required TCP/IP userids for the new TCP/IP system, format the minidisks, and copy the required files from the installed TCP/IP userids to the new TCP/IP userids
3. Configure the new TCP/IP userids
4. Test out the setup
5. Add the new TCP/IP userid to AUTOLOG1

### 1. Add the new system's domain name into the campus name server

Add the new system's domain name to your campus name server. You'll need A, HINFO, and PTR resource records for the new system.

Our installed TCP/IP system is known as VM.MPG.MCGILL.CA. Our new TCP/IP system is known as MUSIC.MPG.MCGILL.CA. The following example shows the resource records for a new system:

```
MUSIC.MPG.MCGILL.CA              A          132.206.131.2
MUSIC.MPG.MCGILL.CA              HINFO      IBM 9370 VM/SP
2.131.206.132.IN-ADDR.ARPA       PTR        MUSIC.MPG.MCGILL.CA
```

You will also need to add new A and PTR records for the installed TCP/IP system which represents the host's IP address for its end of the IUCV link. This is an example of the records required:

```
VM.MPG.MCGILL.CA                 A          132.206.131.1
1.131.206.132.IN-ADDR.ARPA       PTR        VM.MPG.MCGILL.CA
```

**2. Define new TCP/IP userids/Format minidisks/Copy Required Files**

First create the additional userids for the new TCP/IP system. Only the userids for TCPMAINT, TCPIP, SMTP need to be created. MUSIC does not have a TELNET server. MUSIC has its own FTP server so the FTPSERVE virtual machine is not needed. You must also create a ROUTED userid if you are using "dynamic routing" as described in "Routing Notes" above.

Of course, the new userids must have different names from the ones being used in the installed TCP/IP system. Let's use the userids TCPMAIN1, TCPIP1, SMTP1, and ROUTED1 for the new userids. These userids should be defined identical to their counterparts on the installed TCP/IP system. Note that TCPMAIN1 only requires the 191, 591, and 592 minidisks. The 591 minidisk is new for TCP/IP V2R2.

After the userids are created, format all of the new minidisks identical to the equivalent minidisk on the installed TCP/IP system.

Instead of reinstalling TCP/IP on the new system, we'll copy the files we need from the installed TCP/IP system to the new system. We only need the TCPMAINT files, so copy the TCPMAINT 191 minidisk files to TCPMAIN1 191, the TCPMAINT 591 files to TCPMAIN1 591, and the TCPMAINT 592 files to TCPMAIN1 592.

**3. Configure the new TCP/IP userids**

You'll need to make some changes to configure the new TCP/IP system. Both the TCPIP DATA and PROFILE TCPIP files on TCPMAIN1 need to be changed, as well as the PROFILE TCPIP file on TCPMAINT. During the initialization of the TCPIP virtual machine, it first searches for the file "nodename TCPIP", where "nodename" is the node name of the system, as specified by the CMS IDENTIFY command. If this file is not found, the initialization process uses the PROFILE TCPIP file. So wherever you find the PROFILE TCPIP file mentioned in these notes, you should use your "nodename TCPIP" file.

You will also create two files on TCPMAINT that can be used to START and STOP the IUCV link via the OBEYFILE command while you are doing your testing.

a) **Changes to TCPMAIN1's TCPIP DATA file**
For the TCPIP DATA file on TCPMAIN1, change TCPIPUSERID to TCPIP1, and change the HOSTNAME to the name you've decided for this new system. We have chosen the HOSTNAME MUSIC for our new system. We had previously chosen our DOMAINORIGIN to be MPG.MCGILL.CA, so the domain name for our new system is MUSIC.MPG.MCGILL.CA.

b) **Changes to TCPMAIN1's PROFILE TCPIP file**
A number of changes have to be made to TCPMAIN1's PROFILE TCPIP file. First, change the server userids in the AUTOLOG and PORT sections to the new server userids. Comment out those userids in the AUTOLOG section not being used on the new system, and comment out the ports required for those userids not being used in the PORTS section. You will need to change the DEVICE, LINK, HOME and START statements. The following are sample statements:

```
DEVICE IUCVDEV2 IUCV XYZZY XYZZY TCPIP B
LINK IUCVLNK2 IUCV 1 IUCVDEV2
....
HOME
    132.206.131.2 IUCVLNK2
....
START IUCVDEV2
```

If you are using "static routing" as described in "Routing Notes" above, then add the appropriate direct route and DEFAULTNET route in the GATEWAY section to get back to the installed TCP/IP system via the IUCV link using the same subnet mask (if subnetted). The following are sample statements:

```
HOME
; Local host's Internet address
132.206.131.2   IUCVLNK2

GATEWAY
; Network     First hop   Driver  Packet size  Subn mask  Subn value
; Direct routes
132.206.0.0   =          IUCVLNK2 DEFAULTSIZE  0.0.255.0  0.0.131.0
; Default for everything else
DEFAULTNET 132.206.131.1 IUCVLNK2 DEFAULTSIZE  0
```

If you are using "dynamic routing" as described in "Routing Notes" above, then add the appropriate link statement to the BSDROUTINGPARMS statement so that ROUTED can use this information to advertise the reachability of the IUCV link from this end using the same subnet mask (if subnetted). You should only use the statements in the BSDROUTINGPARMS section. All of the statments in the GATEWAY section should be commented out. The following are sample statements:

```
HOME
; Local host's Internet address
132.206.131.2   IUCVLNK2

; ; RouteD Routing information (if you are using the ROUTED server)
; ; If you are using RouteD, uncomment all the lines below for
; ; 'BSDROUTINGPARMS', and comment out all the lines for the 'GATEWAY'
; ; statement.
;
; ;  link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
     IUCVLNK2 1500        0     255.255.255.0   132.206.131.1
ENDBSDROUTINGPARMS
```

c) **Changes to TCPMAINT's PROFILE TCPIP file**

A number of changes have to be made to TCPMAINT's PROFILE TCPIP file for the IUCV link. You will need to add additional DEVICE, LINK, HOME and START statements. The following are prototype statements:

```
DEVICE IUCVDEV1 IUCV XYZZY XYZZY TCPIP1 A
LINK IUCVLNK1 IUCV 1 IUCVDEV1
....
; add the new home definition to the end of the list of the
; other home definitions
HOME
    132.206.131.1 IUCVLNK1
....
```

```
; note this is commented out-we'll start it using obeyfile later
; START IUCVDEV1
```

If you are using "static routing" as described in "Routing Notes" above, then add the appropriate direct route in the GATEWAY section to get to the new subnet(network) via the IUCV link using the same subnet mask (if subnetted). The following are sample statements:

```
HOME
; Local host's Internet addresses
132.206.120.2  ETH1
132.206.131.1  IUCVLNK1

GATEWAY
; Network     First hop   Driver  Packet size  Subn mask  Subn value
; Direct routes
  132.206.0.0 =          IUCVLNK1 DEFAULTSIZE  0.0.255.0   0.0.131.0
  132.206.0.0 =              ETH1 DEFAULTSIZE  0.0.255.0   0.0.120.0
; Default for everything else
  DEFAULTNET 132.206.120.1   ETH1 DEFAULTSIZE  0
```

If you are using "dynamic routing" as described in "Routing Notes" above, then add the appropriate link statement to the BSDROUTINGPARMS statement so that ROUTED can use this information to advertise the reachability of the IUCV link from this end using the same subnet mask (if subnetted). You should only use the statements in the BSDROUTINGPARMS section. All of the statments in the GATEWAY section should be commented out. The following are sample statements:

```
HOME
; Local host's Internet addresses
132.206.120.2  ETH1
132.206.131.1  IUCVLNK1

; ; RouteD Routing information (if you are using the ROUTED server)
; ; If you are using RouteD, uncomment all the lines below for
; ; 'BSDROUTINGPARMS', and comment out all the lines for the 'GATEWAY'
; ; statement.
;
; ;  link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
     ETH1      1500       0     255.255.255.0   0
     IUCVLNK1  1500       0     255.255.255.0   132.206.131.2
ENDBSDROUTINGPARMS
```

d) **Create files on TCPMAINT to START and STOP IUCV link**

During the testing step, you will use the TCPIP OBEYFILE command from TCPMAINT to start and stop the IUCV link. This is a convenient way to test a link without having it affect the operation of the other links. The OBEYFILE command takes a filename as a parameter. The file normally contains statements from the PROFILE TCPIP file. In our case, we will create two files, one to start the IUCV link and the other to stop the link. On TCPMAINT, create the file STARTIUC TCPIP on the 191 minidisk and the file will contain the line

```
START IUCVDEV1
```

Also on TCPMAINT, create the file STOPIUCV TCPIP on 191, and this file will contain the line

```
STOP IUCVDEV1
```

**4. Test out the setup**

When the above configuration step is complete, restart your installed TCP/IP system to get the new definitions in place. After it has been restarted, START the IUCV link on the installed TCP/IP system by logging onto TCPMAINT and issuing the command OBEYFILE STARTIUC. We are using the TCP/IP OBEYFILE command to run the configuration file STARTIUC TCPIP A on TCPMAINT's 191. This file we created in the previous step. Then logoff TCPMAINT.

Note that if any problems arise while you are testing the IUCV link, you can stop the IUCV link by logging onto TCPMAINT and issuing the command OBEYFILE STOPIUCV. The STOPIUCV TCPIP A file used in the OBEYFILE command was created in the previous step.

Next, start the newly installed TCP/IP system, by logging on TCPIP1 and running its profile exec. After it appears to have started okay, disconnect from TCPIP1.

Next logon to TCPMAINT and check the IUCV link using the NETSTAT DEVLINKS command. Also check the gateway configuration using the NETSTAT GATE command. Then logoff TCPMAINT.

Next run NETSTAT DEVLINKS and NETSTAT GATE from TCPMAIN1. Also from TCPMAIN1, try to PING the installed TCP/IP system (by IP number, not hostname, since the name service on the new system may not be working yet) to check that the new system can really talk to the production system. Then check-out whatever other clients you want to on the new system.

Then check out the servers on the new system by logging on to a VM userid that has access to the installed TCP/IP system, and then PING/TELNET/FTP/etc to the new system (by IP number, not hostname, unless you have enrolled the new system in the campus name server). Repeat on another host (PC, UNIX, etc) if desired.

When all of the testing is done and you are satisfied everything works, logon to TCPMAINT and edit PROFILE TCPIP. Remove the semicolon and all of the leading blanks from the start of the START IUCVDEV1 statement so that the START IUCVDEV1 starts in column 1. Now that the tests worked, you'll want this link started automatically. You can also purge the files STARTIUC TCPIP and STOPIUCV TCPIP on TCPMAINT's 191 as they are no longer needed.

**5. Add new TCP/IP userid to AUTOLOG1**

Now that everything is working, you'll want to add TCPIP1 to your AUTOLOG1 profile exec so that the new system is AUTOLOGged when VM is reIPL'd. If you have not added TCPIP to the profile exec prior to now, you should do it now. TCPIP should be autologged before TCPIP1.