INTER-MULTICS FILE TRANSFER FACILITY REFERENCE MANUAL

SUBJECT

Inter-Multics File Transfer (IMFT) Reference Material

SPECIAL INSTRUCTIONS

This edition of the manual supersedes the previous edition CY73-00, dated July 1982 and its addendum CY73-00A, dated February 1983. Throughout this manual, change bars in the margins indicate technical changes and additions, and asterisks denote deletions.

SOFTWARE SUPPORTED

Multics Software Release 10.2

ORDER NUMBER

CY73-01

December 1983

Honeywell

PREFACE

This manual is intended for system administrators, system operators, and users. It contains the information necessary for a system administrator to configure IMFT, a system operator to run IMFT, and a user to issue, delete, list, and move IMFT requests, and display foreign site names.

Section 1 and Section 2 are intended for the system administrator, and contain introductory information and IMFT configuration requirements, respectively.

Section 3 is intended for the system operator and contains the information the system operator might require to run IMFT. Section 4 contains general user information and the commands to issue, delete, list, or move IMFT requests, and list IMFT sites.

Document	Referred to in Text as
Multics Bulk I/O (Order No. CC34)	Bulk I/O
Multics Administrator's Manual Communications (Order No. CC75)	MAM Communications
Multics Administrator's Manual Project (Order No. AK51)	MAM Project

File No.: 1L13, 1U13 CY73-01

Multics Administrator's Manual System Administrator (Order No. AK50)	MAM System
Multics Operator's Handbook (Order No. AM81)	MOH
Multics Programmer's Reference Manual (Order No. AG91)	Programmer's Reference
Multics Commands and Active Functions (Order No. AG92)	Commands
Multics Subroutines and Input/Output Modules (Order No. AG93)	Subroutines

Significant Changes in CY73-01

IMFT is now capable of using an X.25 multiplexed channel to transfer files. A description of how to configure an X.25 channel has been added to the existing HASP information in Section 2.

Section 3 contains updated material on running the IMFT driver processes.

Users can now request a file transfer from a foreign site while logged in to the host site, with the use of the -source control argument to the enter_imft_request command. This command and a new user command, print_imft_sites, are described in Section 4.

Contents

SECTION 1

INTRODUCTION

The Inter-Multics File Transfer Facility (IMFT) lets you transfer files and subtrees between Multics systems. A multiplexed communication channel is required for data transfer. This manual contains a description of IMFT on a HASP multiplexed channel and on an X.25 multiplexed channel.

Before you can use IMFT to transfer files and subtrees between systems, it must be configured as part of both the local system and the remote system. As a system administrator, it is your responsibility to configure IMFT on the system for which you are responsible.

Once configured, IMFT is easily maintained and requires little, if any, operator intervention. User requests for data transfer are placed in priority queues for later processing by an I/O daemon.

STRUCTURE OF AN IMFT CONNECTION

An IMFT connection links two Multics systems, one local and one remote, to permit the transfer of files and subtrees. The IMFT connection requires:

- Two I/O daemon driver processes
- A multiplexed channel (either X.25 or HASP) and appropriate subchannels

This manual provides the information necessary for you, as system administrator, to define the appropriate communication channel and driver processes. For information about hardware requirements and software support for both a HASP multiplexed channel and an X.25 multiplexed channel, see Appendix A of MAM Communications.

I/O Daemon Driver Processes

On both the local system and the remote system, an IMFT connection requires two I/O daemon driver processes:

- Input driver
  receives files and subtrees from the output driver on the remote system

- Output driver
  transmits files and subtrees to the input driver on the remote system, and (optionally) transmits requests for the transfer of files and subtrees from the output driver of the remote system

Associated with the output driver on each system is a set of priority queues in which users place transfer requests using the enter_imft_request command. These queues are defined with a single I/O daemon Request_type, as described below. There may also be a set of queues used to request transfers from the remote system; these queues are defined with a separate Request_type.

For a complete description of I/O daemons, refer to the Bulk I/O manual.

Communication Channels

Each IMFT configuration requires either a HASP multiplexed channel or an X.25 multiplexed channel. There is a major difference between the two channel types. An X.25 channel operates with subchannels that can both send **and** receive information. A data channel that functions in this manner is called bidirectional. A HASP channel, on the other hand, has subchannels that either send **or** receive data; these are unidirectional channels that are linked into pairs, as described below.

HASP MULTIPLEXED CHANNEL AND SUBCHANNELS

An IMFT configuration on a HASP multiplexed channel requires the following subchannels:

- 1 operator subchannel
- 2 card-reader subchannels
- 2 card-punch subchannels

An IMFT connection uses the card-reader and card-punch subchannels of a HASP multiplexed channel to allow simultaneous data transfer in both directions over a single physical channel. Although IMFT does not actually use physical card-readers or card-punches, a HASP multiplexed channel permits its subchannels to be configured as operator consoles, card-readers, card punches, or line printers. IMFT uses a card-reader and card-punch subchannel pair to simulate an ordinary bidirectional data channel. Each driver requires its own pair of subchannels and, therefore, an IMFT connection uses two card-reader subchannels and two card-punch subchannels.

The following figure shows a HASP configuration for both System A and System B. "pun" means card punch, and "rdr" means card reader.

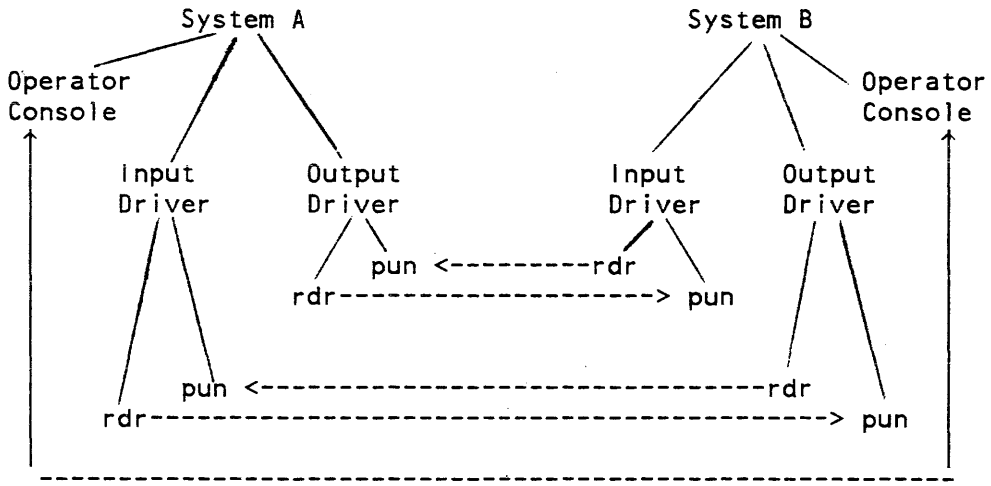

Figure 1-1. Hasp Configuration

X.25 MULTIPLEXED CHANNEL AND SUBCHANNELS

An IMFT configuration on an X.25 multiplexed channel requires the following subchannels:

- 1 slave subchannel
- 1 autocall subchannel

An X.25 connection is bidirectional; it is configured as an autocall subchannel on one end (at System A), and as a slave subchannel on the other end (at System B). Both subchannels can receive or send data. The autocall or dialout subchannel on one system initiates the connection with the corresponding slave subchannel on the other system.

The following figure shows the X.25 IMFT configuration between System A and System B. "slv" stands for slave subchannel, and "auto" stands for autocall subchannel.

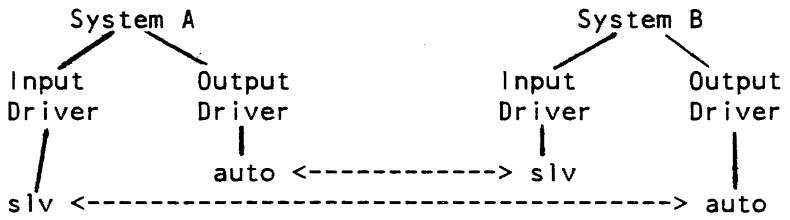

Figure 1-2. X.25 Configuration

In the above figure, the fact that both systems have one slave subchannel and one autocall subchannel is arbitrary. The X.25 configuration is equally valid if System A has two slave subchannels and System B has two autocall subchannels, or vice versa. What is important is that each connection is configured as a slave subchannel on one end, and an autocall subchannel on the other.

Connection Types

The physical connection between two systems is either a hardwired connection or a dial-up connection. In a hardwired connection, the two systems are permanently connected via dedicated phone lines or other similar equipment. In a dial-up connection, one system (system operator) makes a phone call to the other to establish the connection. The IMFT connection can be either hardwired or dial-up. Connection type must be either a Binary Synchronous Communication (BSC) line (for a HASP connection), or a High-level Data Link Control (HDLC) line (for an X.25 connection), with line speeds up to 72kb.

If you use a hardwired connection, you can define the channel and driver processes to permit automatic operation of the connection without operator intervention. If you use a dial-up connection, some operator intervention is required to start and stop the driver processes on both systems.

SECTION 2

ADMINISTRATION OF AN IMFT CONNECTION

To configure an IMFT connection, you must:

1. Define the HASP or X.25 multiplexed channel.
2. Define the I/O daemons, which includes the following steps:
   - Determine the Person_id and Project_id names, as well as the local system name and the foreign system name(s)
   - Decide on access categories and levels, if applicable
   - Define the devices and the Request_types in the I/O daemon tables
   - Register the I/O daemon in the Project Master File (PMF) and in the Person Name Table (PNT)
3. Initialize IMFT for operation

This section contains the information necessary to perform all of the above procedures.

DEFINING A MULTIPLEXED CHANNEL

IMFT requires a HASP or X.25 multiplexed channel. You must define this channel, like all communications channels, in the Channel Master File (CMF). IMFT requires a definition for the communications channel on each system.

If you are setting up a HASP channel, configure it on one system as the "host"; on the other system configure the channel as the "workstation". Either system can be the host or the workstation. The terms host and workstation do not have their usual meaning here. Ordinarily a system configured as a HASP workstation performs remote-job-entry (RJE) tasks; it is usually an RJE terminal with an operator's console and one or more card readers, line printers, and card punches. Job decks are transmitted to the HASP host system for execution, and sent to the workstation for printing/punching or online perusal. However, in an IMFT configuration, each system performs as **both** host and workstation; each system can both send and receive file transfers. Therefore, although one system is called the host and the other the workstation, the two systems do not differ greatly in their IMFT capabilities.

If you are setting up an X.25 channel, configure it on one system as the "DCE", and on the other system configure it as the "DTE". Either system can be the DCE or the DTE. DCE stands for Data Communications Equipment, and DTE means Data Terminal Equipment. For an IMFT configuration, both the DTE system and the DCE system have the ability to send and receive file transfers.

For both HASP and IMFT, specify the channel's configuration parameters by terminal types defined in each system's Terminal Type Files (TTF).

Terminal Type File

The Terminal Type File (TTF) defines all terminal types known to the system. It is not the intent of this section to define all aspects of the TTF, but only the information necessary to define the IMFT connection. For a complete description of the TTF, refer to the Programmers' Reference manual.

The pathname of the TTF is usually:

>udd>SysAdmin>admin>TTF

Once you have made the appropriate entries to the TTF, compile it via the cv_ttf command (see the Programmer's Reference manual) and install it via the install command (see MAM - Project).

TTF ENTRIES FOR AN IMFT HASP CHANNEL

IMFT requires the following TTF entries for the HASP multiplexed channels:

terminal_type: <type_name>;

The <terminal_type> describes the characteristics of the **remote** system. For IMFT, the <terminal_type> name is variable; one of the following names is recommended:

- IMFT_HASP_HOST
  if the system is the **workstation**
- IMFT_HASP_WORKSTATION
  if the system is the **host**

additional_info: <string>;

The additional_info statement specifies the additional information needed to run the terminal on an IMFT connection. The additional_info arguments required for IMFT are: type= host (type= workstation for a system configured as the host), and rts_mode= no. If you are running IMFT without operator intervention (see mode= auto in "Device Definition for IMFT"), it is recommended that you specify connect_timeout= none. For a detailed description of these arguments see Appendix A of MAM Communications.

Example 1

```
terminal_type: IMFT_HASP_HOST;
 additional_info: "type= host, connect_timeout= none,
                   rts_mode= no"
```

In the above example, a terminal type is defined for a system configured as the workstation.

Example 2

```
terminal_type: IMFT_HASP_WORKSTATION;
 additional_info: "type= workstation, connect_timeout= none,
                   rts_mode= no"
```

In the above example, a terminal type is defined for a system configured as the host.

TTF ENTRIES FOR AN IMFT X.25 CHANNEL

IMFT requires the following TTF entries for the X.25 multiplexed channels:

terminal_type: <type_name>;

The <terminal_type> describes the characteristics of the **local** system; in this case, it describes whether the system functions as the DTE or the DCE. In setting up an IMFT configuration on X.25 between two Multics systems, it doesn't matter which system is configured as the DTE and which is configured as the DCE. The IMFT <terminal_type> name is variable; one of the following names is recommended:

- IMFT_X25_DCE
  if the system is the DCE
- IMFT_X25_DTE
  if the system is the DTE

additional_info: <string>;

The additional_info statement specifies the additional information needed to run the terminal on an IMFT connection. The only additional_info arguments required for an IMFT X.25 multiplexer are: type= and n_lc=. The argument n_lc= has no default value. This parameter specifies the number of X.25 logical channels to be defined. It can be any number between 1 and 4095. The type= argument has a value of DTE or DCE; it specifies whether the Multics system behaves as the DTE or the DCE. The terminal_type definitions for both multiplexers (site A and site B) must be such that one is a DTE and the other is a DCE; as stated above, it does not matter which site is configured as the DCE, and which site is configured as the DTE. (For example, if site A's terminal type is "type=DTE", site B's terminal type must include "type=DCE", or vice versa.) For a detailed description of the optional parameters included in the additional_info statement, see MAM Communications.

Example 1

```
terminal_type: IMFT_X25_DCE;
 additional_info: "n_lc=32 type=DCE"
```

In the above example, a terminal type is defined for a system configured as a DCE.

Example 2

```
terminal_type: IMFT_X25_DTE
 additional_info: "n_lc=20 type=DTE"
```

In the above example, a terminal type is defined for a system configured as a DTE.

Channel Master File

You must define a HASP or X.25 multiplexed channel in the CMF. The Channel Master File (CMF) contains the description of all Front-end Network Processors (FNPs), multiplexers, and terminal channels in a system. It is not the intention of this section to detail the CMF definition in its entirety, but only the information necessary to define the IMFT connection. You should be able to edit the IMFT specific configuration information into an existing CMF. Refer to the MAM Communications manual for a full description of CMF creation; Appendix A describes CMF entries for a HASP multiplexed channel and for an X.25 multiplexed channel.

If you are using a hardwired connection, the channel's definition in the CMF on both systems should specify that the multiplexer is "active". Additionally, for both systems specify "connect_timeout= none" as one of the channel's configuration parameters. By including these specifications for a hardwired connection, the channel is always ready whenever both systems and both Front-end Network Processors (FNP) to which the channel is connected are running.

The pathname of the CMF is usually:

>udd>SysAdmin>admin>CMF

Once you have made the required entries to the CMF file:

1. Convert the CMF file to the required Channel Definition Table (CDT) via the cv_cmf command (see the MAM Communications manual)

2. Install the converted version of the CMF via the install command (see MAM Project manual)

CMF ENTRIES FOR A HASP MULTIPLEXED CHANNEL

The HASP channel consists of:

- One operator terminal channel
- At least two card-reader subchannels
- At least two card-punch subchannels

and the other required CMF entries for the HASP channel.

IMFT requires the following CMF entries for the HASP multiplexed channel:

name: <channel_name>;
: Specifies a unique channel name for the HASP channel. The channel name must be of the form x.hNNN where x is a unique letter from a through h identifying the Front-end Network Processor (FNP); hN is the High Speed Line Adaptor (HSLA) identifier (h0, h1, or h2); NN is a 2-digit channel number identifying a subchannel of the specified HSLA (see the Programmer's Reference manual).

service: <service_type>;
: IMFT requires the <service_type> to be "multiplexer" for the HASP multiplexed channel.

multiplexer_type: <type>, <active or inactive>;
: The <type> must be hasp; for a hardwired connection, it is recommended that "active" also be specified (or omitted since it is the default); for a dial-up connection, active or inactive can be used.

baud: <baud_rate>;
: Specifies the baud rate of the channel. The <baud_rate> must be compatible with the FNP line adapter type and be chosen from the following list:

2400	19200
4800	40800
7200	50000
9600	72000

line_type: <line_type>;
: For IMFT, the <line_type> must be "BSC".

terminal_type: <terminal_type>;

For IMFT, the <terminal_type> is variable; one of the following names is recommended:

- IMFT_HASP_HOST
  if this system is being configured as a workstation.

- IMFT_HASP_WORKSTATION
  if this system is being configured as a host.

Example

```
name: b.h203;
   service: multiplexer;
   multiplexer_type: hasp;
   baud: 4800;
   line_type: BSC;
   terminal_type: IMFT_HASP_HOST
```

In the above example, a CMF entry is created for a HASP multiplexed channel. The system is configured as the workstation. The <terminal_type> tells your local system how the **remote** system is defined.

CMF Entries for a HASP Operator Terminal

IMFT uses the HASP subchannels: operator, card-reader, and card-punch. To define the operator subchannel of the HASP multiplexed channel, the following arguments are required:

name: <channel_name>;

Although a pair of names can be specified in this name statement, the operator channel requires only the name of the channel number specified for the unique HASP channel name referred to in "CMF ENTRIES FOR HASP MULTIPLEXED CHANNEL" and the suffix: .opr.

service: <service_type>;

<service_type> must be specified as "slave", designating that the service is used only for special channels attached by the I/O daemons for bulk data input/output or other special subsystems.

line_type: <line_type>;

For IMFT, the <line_type> must be "BSC".

Example

```
name: b.h203.opr;
service: slave;
line_type: BSC
```

In the above example, a CMF entry is created for a local system designated as the workstation.

CMF Entries for Card-Reader Subchannels

IMFT requires that you define at least two card-reader subchannels in the CMF. The card-reader subchannels require the following CMF entries:

name: <channel_name 1> - <channel_name 2>;
: For the card-reader subchannels of the HASP multiplexed channel specify x.hNNN.rdrn - x.hNNN.rdrn where x.hNNN is the unique HASP channel number specified above. Up to eight reader subchannels can be defined. IMFT requires any two card-reader subchannels.

service: <service_type>;
: The <service_type> must be specified as "slave", designating that the service is used only for special channels attached by the I/O daemons for bulk data input/output or other special subsystems.

line_type: <line_type>;
: For IMFT, the <line_type> must be "BSC".

Example

```
name: b.h203.rdr1 - b.h203.rdr8;
service: slave;
line_type: BSC;
```

In the above example, eight card-reader subchannels are defined. IMFT requires two reader subchannels.

CMF Entries for Card-Punch Subchannels

IMFT requires that you define at least two card-punch subchannels of the HASP channel in the CMF. The card-punch subchannels require the following CMF entries:

name: <channel_name 1> - <channel_name 2>;
: For the card-punch subchannels, specify x.hNNN.punn - x.hNNN.punn where x.hNNN is the unique HASP channel number specified above. The total number of card punches and line printers configured cannot exceed 8. IMFT requires any two card-punch subchannels.

service: <service_type>;

The <service_type> must be specified as "slave", designating that the service is used only for special channels attached by the I/O daemons for bulk data input/output or other special subsystems.

line_type: <line_type>;

For IMFT, the <line_type> must be "BSC".

Example

```
name: b.h203.pun1 - b.h203.pun8;
service: slave;
line_type: BSC;
```

In the above example, eight card-punch subchannels are defined. IMFT requires two card-punch subchannels.

Channel Master File Examples for IMFT HASP Configuration

Example 1

Example 1 is a sample CMF file entry for an IMFT HASP connection on a local system configured as the workstation.

```
name: b.h203;
  service: multiplexer;  multiplexer_type: hasp;  baud: 4800;
           line_type: BSC;  terminal_type: IMFT_HASP_HOST;
   name: b.h203.opr;  service: slave;  line_type: BSC;
   name: b.h203.rdr1-b.h203.rdr8;  service: slave; line_type: BSC;
   name: b.h203.pun1-b.h203.pun8;  service: slave; line_type: BSC;
```

Example 2

The following example is a sample CMF file for an IMFT connection on a remote system configured as the host:

```
name: c.h208;
  service: multiplexer;  multiplexer_type: hasp;  baud: 9600;
           line_type: BSC;  terminal_type: IMFT_HASP_WORKSTATION;
   name: c.h208.opr;  service: slave;  line_type; BSC;
   name: c.h208.rdr1-c.h208.rdr8;  service: slave; line_type: BSC;
   name: c.h208.pun1-c.h208.pun8;  service: slave; line_type: BSC;
```

CMF ENTRIES FOR AN X.25 MULTIPLEXED CHANNEL

You must define an X.25 multiplexed channel in the CMF. The X.25 channel uses the following subchannels:

- At least one slave (inward) subchannel
- At least one autocall (outward) subchannel

and other required CMF entries for the X.25 channel.

Although the X.25 channel must be configured as a slave subchannel on one end and as an autocall subchannel on the other end, this does not mean that two systems with an IMFT X.25 connection must each have a slave subchannel and an autocall subchannel. It is equally feasible for System A to have two autocall subchannels, and System B to have two slave subchannels, or vice versa.

The following CMF entries for the X.25 multiplexed channel must be defined:

name: <channel_name>;
Specifies a unique channel name for the X.25 channel. The channel name must be of the form x.hNNN, where x is a unique letter from a through h identifying the Front-End Network Processor (FNP); hN is the High Speed Line Adaptor (HLSA) identifier (h0, h1, or h2); and NN is a 2-digit channel number identifying a subchannel of the specified HSLA (see the Programmer's Reference manual).

service: <service_type>;
IMFT requires the <service_type> to be "multiplexer" for the X.25 multiplexed channel.

multiplexer_type: <type>, <active or inactive>;
The <type> must be X25. For a hardwired connection, it is recommended that "active" also be specified (or omitted if it is the default). For a dial-up connection, active or inactive can be used.

baud: <baud_rate>;
Specifies the baud rate of the channel. The <baud_rate> must be compatible with the FNP line adapter type and be chosen from the following list:

2400	19200
4800	40800
7200	50000
9600	72000

line_type: <line_type>;
For IMFT, the <line_type> must be "X25LAP".

terminal_type: <terminal_type>;

The <terminal_type> describes whether the system functions as the DTE or the DCE. In setting up an IMFT configuration on X.25 between two Multics systems, it doesn't matter which system is configured as the DTE and which is configured as the DCE. The IMFT <terminal_type> name is variable; one of the following names is recommended:

- IMFT_X25_DCE
  if this system is being configured as a DCE.

- IMFT_X25_DTE
  if this system is being configured as a DTE.

Example

In the example below, a CMF entry is shown for an X.25 multiplexed channel. The system is configured as the DCE.

```
name: g.h026;
  service: multiplexer;
  multiplexer_type: x25;
  baud: 9600;
  line_type: X25LAP;
  terminal_type: IMFT_X25_DCE;
```

There are other statements that may be included in an X.25 channel description in the CMF. See MAM Communications for a full description of these optional statements.

CMF Entries for X.25 Slave Subchannels

A slave subchannel is one which is attached to the system by another process; that process then manages the slave channel. To define the slave subchannel of the X.25 multiplexed channel, the following arguments are required:

name: <channel_name>;

For the slave subchannel, specify x.hNNN.slvnn, where x.hNNN is the unique X.25 channel name referred to in "CMF Entries for an X.25 Multiplexed Channel". A pair of names can be specified in this name statement.

service: <service_type>;

The <service_type> must be "slave", designating that the service is only for special channels attached by the I/O daemons for bulk data input/output or other special subsystems.

Example

```
name: g.h026.slv01 - g.h026.slv02;
service: slave;
comment: "X.25 slave channels for IMFT input driver";
```

In the above example, two slave subchannels are defined. IMFT requires a minimum of one slave subchannel. Note that this CMF entry contains the optional statement "comment:". For a description of optional CMF statements, see MAM Communications.

CMF Entries for X.25 Autocall Subchannels

An autocall or dialout subchannel is a process that can initiate an outbound connection with another process (i.e., a slave terminal). To define the autocall subchannel of the X.25 multiplexed channel, the following arguments are required:

name: <channel_name>;
: For the autocall subchannel, specify x.hNNN.autnn - x.hNNN.autnn, where x.hNNN is the unique X.25 channel name referred to in "CMF Entries for an X.25 Multiplexed Channel".

service: <service_type>;
: The <service_type> must be "autocall" for IMFT.

Example

```
name: g.h026.aut01 - g.h026.aut04;
service: autocall;
```

In the above example, four autocall subchannels are defined. IMFT requires a minimum of one autocall subchannel.

Channel Master File Example for IMFT X.25 Configuration

Example 1 is a sample CMF file entry for an IMFT X.25 connection on a local system configured as a DCE.

Example 1

```
name: g.h026;
  service: multiplexer;  multiplexer_type: x25;  baud: 9600;
           line_type: X25LAP;  terminal_type: IMFT_X25_DCE;
   name: g.h026.slv01 - g.h026.slv04;  service: slave;
   name: g.h026.aut05 - g.h026.aut11;  service: autocall;
```

Example 2

The following example is a sample CMF file for an IMFT X.25 connection on a remote system configured as a DTE:

```
name: b.h001;
  service: multiplexer;  multiplexer_type: x25;  baud: 9600;
           line_type: X25LAP;  terminal_type: IMFT_X25_DTE;
   name: b.h001.slv01 - b.h001.slv04;  service: slave;
   name: b.h001.aut05 - b.h001.aut11;  service: autocall;
```

DEFINING THE I/O DAEMON FOR IMFT

To define the daemon for IMFT:

1. Add the definitions of the input and output drivers, and the definition(s) of the Request_type(s) used by the user commands to the I/O daemon tables.

2. Register the I/O daemon in the Project Master File (PMF) and in the Person Name Table (PNT).

Prior to making the entries in these special files, review the information required in the sections below, and decide on the Person_id and Project_id names you want to assign to IMFT, as well as the names you want to assign to the local system and the foreign system. If you are using the Access Isolation Mechanism (AIM), decide upon the access levels and categories you plan to assign.

Choosing Names for IMFT Variables

Prior to adding the required IMFT entries to the appropriate system files and tables, choose the names you want to assign to variable entries such as:

- Person_id
- Project_id
- local_system name
- foreign_system name

For the Person_id it is recommended that you use IMFT.

Every project on the system has a Project_id in the Project Master File (PMF). A separate Project_id can be created for IMFT, or you can add keywords and their values in the PMF placed under an existing Project_id. If IMFT is to be available on a system wide basis, add the required PMF entries to a system overhead project such as "Daemon" or "SysDaemon". If IMFT is to be available for a small group of users in a specific project, add the required entries to that project entry in the PMF.

When choosing a local system and foreign system identifier, the local_system keyword on one system must be the same as the foreign_system keyword on the other system, and the foreign_system keyword on one system must be the same as the local_system keyword on the other system. The foreign_system name must be used to generate the Request_type <name>s, as described later in this section.

Access Isolation Mechanism Considerations

IMFT is designed to interact with the Access Isolation Mechanism (AIM) provided that the proper sensitivity level and access categories are specified in the max_access_class statement (see "IMFT Request Type Definition" below).

When using AIM with IMFT, the AIM sensitivity levels and the access categories must have the same meanings on both systems. When specifying the sensitivity level, specify the highest level whose meaning is the same on both systems. Data equal to or less than this level can then be transferred.

Not only must the access categories have the same meanings on both systems, but they must be specified in the same positions (order) on both systems.

For example, here is the order of access categories for System 1 and System 2:

System 1	System 2
lisd	dscc
sstd	sstd
mslc	lisd

For these two systems, only the sstd category can be transferred. Even though the lisd category exists on both systems, it occupies different positions on each system, so it is unusable.

Files and subtrees can be transferred only if their access class is less than or equal to (1) the process authorization of the user who submitted the request and (2) the common access class ceiling. See "Common Access Class Ceiling" in Section 4 for a definition of the common access class ceiling.

Files and subtrees are created on the foreign system with the same access class that they have on the local system.

The max_access_class keyword (described in "Device Definition for IMFT" below) can be used in the args statements of the input and output drivers to restrict the level of data transfer to an access class less than the common access class ceiling. If this keyword is used, it must be specified in both the output driver on one system and the corresponding input driver on the other system.

If the max_access_class keyword is used, create the driver processes on both systems with a process authorization that is greater than or equal to the access class specified by the keyword. Otherwise, the processes must be created with an authorization that is greater than or equal to the common access class ceiling. In neither case do the process authorization of the output driver and the corresponding input driver on the remote system need to be equal. See "IMFT Person Name Table Entries" later in this section for a discussion of how to set the process authorizations of the driver processes.

The min_access_class keyword (described in "Device Definitions for IMFT" below) can be used in the args statement of the input and output drivers to restrict the level of data transfer to an access class above the specified minimum. If this keyword is used, it must be specified in both the output driver on one system, and the coresponding input driver on the other system.

All IMFT driver processes must be given access to the system_privilege_ gate. See "system_privilege_ Access" later in this section for a description of how to insure that this access is granted.

For a complete description of AIM, refer to the Programmer's Reference manual.

I/O Daemon Table

I/O daemon tables define the devices and Request_types to be used with the I/O daemon. A source file consists of a sequence of statements and substatements that define and describe each device and Request_type. It is not the intent of this section to present a full description of I/O daemon tables, but only the device and Request_type definitions required for IMFT I/O daemon definition. For a full description of I/O daemon tables, refer to the Bulk I/O manual.

The pathname of the source of the I/O daemon tables is usually:

>ddd>idd>iod_tables.iodt

Once you have edited the appropriate information into the I/O daemon tables source, the tables must be compiled via the iod_tables_compiler command (see the Bulk I/O Manual). It is recommended, for convenience, that the pre-compiled version of the I/O daemon tables be stored in the same directory as the compiled version with the name iod_tables.iodt.

I/O DAEMON DEVICE DEFINITION FOR IMFT

The IMFT driver requires that you define two major devices in the I/O daemon tables: one for the input driver and one for the output driver. These devices must specify use of the "imft_driver_" driver module.

You must define at least one minor device for the output driver of each site, and you may define a maximum of two. Before choosing minor devices, decide whether you want to be able to:

- transfer files to the remote site
- transfer files from the remote site
- perform both of the above functions

If you only want to transfer files to the remote site, you must define one output driver minor device that enables such transfers. Likewise, if you only want to request file transfers from the remote site, you must define one output minor device that enables this type of transfers. If you want to be able to transfer files to and from the remote site, you must define two minor devices: one enabling transfers from the foreign site, and the other enabling transfers to the foreign site. Both of these minor devices are described later in this section.

The IMFT driver does not support the "line: variable;" construct. Additionally, the HASP or X.25 subchannels used by a driver are specified in the args statement. Therefore, the line statement used for the IMFT driver must be "line: *;".

The args statement specifies the following:

- direction of transfer for the driver
- the "attach" descriptions for the subchannels used by the driver
- the Person_ids used to validate the local and remote systems
- whether or not transfers are initiated automatically when the physical connection is established

If a hardwired connection is used and the channel is configured as described above, it is recommended that all four I/O daemon driver processes for the connection be logged in automatically by either the system_start_up.ec or the start_up.ec executed by Utility.SysDaemon. Additionally, it is recommended that all four drivers specify "mode= automatic" in their respective args statements. By including these specifications, the IMFT connection will run automatically without operator intervention whenever both systems are running.

Device Statements

IMFT requires the following device statement and substatements to define the input and output driver in the I/O daemon tables:

Device: <name>;

Defines the name of a major device and denotes the beginning of a device description. Any subsequent substatements (see below) apply to this device until the next Line, Device, or Request_type statement is encountered. Any <name> can be chosen; it can be a maximum of 24 characters and cannot contain periods or spaces.

driver_module: <name>;

For IMFT, <name> must be "imft_driver_".

line: <name>;

For IMFT, <name> must be "*".

args: <string>;

Defines the characteristics of this device. <string> is a quoted string consisting of a series of keyword/value pairs separated by commas. The syntax of a keyword/value pair is:

keyword= value

No space is permitted after the keyword and before the equal sign. If the value contains spaces, commas, quotes, or equal signs, it must be quoted.

Define the following keyword/value pairs for IMFT:

direction= <input/output>

Specifies whether this is an input or output driver. <input/output> must be either "input" or "output".

local_system= <Person_id>

Specifies the name of the local system. This name is used to validate the connection. See "Access Isolation Mechanism Considerations" above for more information. This keyword is required.

foreign_system= <Person_id>

Specifies the Person_id of the remote system. This Person_id is also used to validate the connection. See "Access Isolation Mechanism Considerations" above for more information. This keyword is required and must also be used in constructing the Request_type names.

input_description= <quoted_string>

ids= <quoted_string>

Specifies the attach description for the input subchannel for this driver. Note that a quoted string is required, thereby creating an entry with double quotes (see example). This keyword is required if the I/O module in use is either hasp_host_ or hasp_workstation_. It is not appropriate for the subchannels of an X.25 connection.

output_description= <quoted_string>
ods= <quoted_string>

Specifies the attach description for the output subchannel for this driver. Note that a quoted string is required, thereby creating an entry with double quotes (see example). This keyword is required if the I/O module in use is either hasp_host_ or hasp_workstation_. The keyword is not appropriate for the subchannels of an X.25 connection.

io_description= <quoted_string>
iods= <quoted_string>

specifies the attach description for the channel used by this driver for both input and output. Note that a quoted string is required, thereby creating an entry with double quotes (see example). This keyword is required if the communications protocol in use is an X.25 connection. It is not appropriate if the I/O module is hasp_host or hasp_workstation.

mode= <auto/manual>

Specifies whether this driver is to operate with or without operator intervention. <auto/manual> must be either "automatic", "auto", or "manual". (The value "auto" is a short form of "automatic".) This keyword is optional and defaults to "manual". Use of "auto" or "automatic" implies either "auto_receive= yes" or "auto_go= yes" as appropriate, causes the driver to wait indefinitely for completion of the connection sequence, and causes the driver to wait for the remote system's driver to reconnect again whenever the remote driver disconnects. Use of "manual" implies either "auto_receive= no" or "auto_go= no" as appropriate, causes the driver to wait no more than five minutes for completion of the connection sequence, and causes the driver to logout whenever the remote system's driver disconnects.

auto_go= <yes/no>

Specifies whether an output driver should immediately begin to transmit files and subtrees to the remote system or should instead wait for an operator command after the connection is established. <yes/no> must be either "yes" or "no". This keyword cannot be specified for an input driver. This keyword is optional and defaults to "no" if "mode= manual" is specified and defaults to "yes" if "mode= automatic" is specified.

auto_receive= <yes/no>

Specifies whether an input driver should immediately wait for files and subtrees from the remote system, or should wait for an operator command after the connection is established. <yes/no> must be either "yes" or "no". This keyword may not be specified for an output driver. This keyword is optional and defaults to "no" if "mode= manual" is specified and defaults to "yes" if "mode= automatic" is specified.

allow_remote_request= <yes/no>

Specifies whether or not an input driver accepts requests from the remote system for the transfer of files from the local system. <yes/no> must be either "yes" or "no". This keyword is optional and defaults to "no". It can not be specified for an output driver.

explicit_access= <yes/no>

Specifies whether an explicit ACL term is required for requests for remote transfer. <yes/no> must be either "yes" or "no". If the keyword is "yes", then if a request is made at the remote system for a file transfer from the local system to the remote system, the transfer cannot occur unless the Person_id of the output driver on the local system appears explicitly in the ACL of the file. This keyword is optional and defaults to "yes". It can not be specified for an output driver.

max_access_class= <quoted_string>

Specifies the maximum access class for data that may be transferred across this connection. The access class specified must be less than or equal to the common access class ceiling between the two systems (as defined in Section 4 of this manual). If given, this keyword must be specified for both the output driver on one system and the corresponding input driver on the other system. If not given, the common access class ceiling is used as the limit for data transfer.

min_access_class= <quoted string>=

Specifies the minimum access class for data that may be transferred across this connection. The access class must be less than or equal to the common access claass calling between the two systems (as defined in Section 4 of this manual). If both min_access_class and max_access_class are given, the min_access_class must be less than or equal to the max_access_class. If the keyword is given, it must be specified for both the output driver on one system and the corresponding input driver on the other system. If it is not given, a minimum access class of "system_low" is used.

version= <old/new>

Specifies whether the remote system is running an old or new version of IMFT. The old version is the one provided in MR10.1, and a new version is one later than MR10.1. This keyword is optional; the default is "new". In order to communicate with the old version, the keyword and a value of "old" must be supplied.

indirect= <path>

Specifies a segment or archive component containing the keyword/value pairs for this device. <path> is the pathname of the segment or (if the :: convention is used) the archive component. This keyword must be used if the length of the args string is greater than 256 characters. If it is used, no other keyword should be included in the args statement; they should all appear in the specified segment or archive component. In this case, the entire args string is not quoted, and the quotes in the input_description, output_description, or io_description values are not doubled.

minor_device: <name>;

Defines the name of a minor device of an output driver, and denotes the beginning of a minor device description. (You must define at least one output minor device for each IMFT site, and you may define two. Your choices are one or both of the following minor devices: one that enables file transfers to the foreign site, and one that enables files to be transferred from the foreign site.) All subsequent substatements (see example) refer to this minor device until the occurrence of the next Line, Device, minor_device or Request_type statement. <name> can be any string of up to 24 characters; it cannot contain periods or spaces. The default_type substatement is required for each minor device.

default_type: <name>;

Identifies the Request_type serviced by a device or minor device. It refers to a device for an input driver, and a minor device for an output driver. <name> must be the same as the Person_id of the foreign system, prefaced by either "From_" or "To_" (e.g., From_System_M). <name> must be the same as that of the Request_type that identifies this device or minor device.

Example 1

The following example defines an IMFT output driver on a HASP multiplexed channel. Note that two minor devices are defined, which enables the output driver to transfer files to the remote site, and request transfers from the remote site.

```
Device:              system_m_ft_out;
 driver_module:        imft_driver_;
 line:                 *;
 args:                 "direction=output, local_system=MIT,
                          foreign_system=System-M
                          ods=""hasp_host_ -comm hasp
                                -tty b.h203.rdr1 -device reader"",
                          ids=""hasp_host_ -comm hasp
                                -tty b.h203.pun1 -device punch"",
                          auto_go=yes";
minor_device:          to;
  default_type:          To_System-M;
minor_device:          from;
  default_type:          From_System-M;
```

Example 2

The example below defines an IMFT input driver on a HASP multiplexed channel.

```
Device:              mit_file_transfer_in;
  driver_module:       imft_driver_;
  line:                *;
  args:                "direction=input, local_system=System-M,
                          foreign_system=MIT,
                          ids=""hasp_workstation_ -comm hasp
                                -tty b.h203.rdr2 -device reader"",
                          ods=""hasp_workstation_ -comm hasp
                                -tty b.h203.pun2 -device punch"",
                          allow_remote_request=yes,
                          explicit_access=no,
                          auto_receive=yes";
  default_type:        To_MIT;
```

The next set of examples define IMFT drivers on an X.25 channel. Note that both examples contain the io_description= (iods=) keyword, which is used for bidirectional channels such as X.25, in contrast to the input_description= (ids=) keyword and the output_description= (ods=) keyword, which are used for unidirectional channels such as HASP.

Example 3

In the following example, a device is defined for an IMFT output driver on an X.25 connection. Note that only one minor device is defined; in this case, the output driver can only transfer files to the remote site.

```
Device:              cisl_file_transfer_out;
  driver_module:       imft_driver_;
  line:                *;
  args:                "direction=output, local_system=System-M,
                          foreign_system=CISL,
                          max_access_class=system_low,
                          iods=""tty_ b.h001.*,
                          -ds 31060850:slv01"", mode=automatic";
minor_device:        to;
  default_type:        To_CISL;
```

Example 4

The following example defines an IMFT input driver on an X.25 connection.

```
Device:              phx_file_transfer_in;
  driver_module:       imft_driver_;
  line:                *;
  args:                "direction=input, local_system=CISL,
                          foreign_system=System-M,
                          max_access_class=system_low,
                          iods=""tty_ b.h001.slv01"",
                          mode=automatic";
  default_type:      To_System-M;
```

REQUEST_TYPE DEFINITION FOR IMFT

You must specify a Request_type in the manner described below, for each device that you defined in the I/O daemon tables.

If you defined only one minor output device and it is the device that enables transfers to the remote site, you must define one Request_type that is for transfers to the foreign site, and specify the appropriate minor output device. Specify the input driver in this Request_type, as well. The name of this Request_type description must be the same as the foreign system ID specified in the args statement of the drivers, prefaced by the string "To_" (e.g., To_MIT).

If you defined only one minor output device and it is the device that enables transfers from the remote site, you must define one Request_type that is for transfers from the foreign site, and specify the appropriate minor output device. Specify the input driver in this Request_type description as well. The name of this Request_type must be the same as the foreign system ID specified in the args statement of the drivers, prefaced by the string "From_" (e.g., From_MIT).

If you defined two minor output devices for IMFT, you must define two Request_types: one for transfers to the foreign site, specifying the appropriate minor device, and the other for transfers from the foreign site, mentioning the appropriate minor device. The input driver must be specified in one (and only one) of the two Request_type descriptions, as well.

Currently, the IMFT facility does not charge users for use of the facility. Therefore, the Request_type defined for an IMFT driver must include the statement "accounting: nothing;".

If the Access Isolation Mechanism (AIM) is enabled, to ensure proper operation of the daemon the definitions of the Request_types used for the input and output drivers must include the statement:

max_access_class: system_high;

If this statement is omitted, the coordinator will leave requests in the queue indefinitely. See "Access Isolation Mechanism Considerations" above for more detail.

By default, the IMFT user commands use the Request_type "imft". If you wish to define a default remote system for transfer requests, define the "imft" Request_type in the I/O daemon tables with the same values specified for the driver_userid, default_queue, max_queues, max_access_class, and device statements as are specified for the actual Request_type for that remote system. In addition, the following commands should be issued after using the create_daemon_queues command to make the "imft" Request_type a synonym of the actual request type:

```
delete imft_*.ms
add_name To_Site-Name_(1 2 ... N).ms imft_(1 2 ... N).ms
```

where To_Site-Name is the Request_type name as given in the I/O daemon tables and N -4 is the number of queues defined for that Request_type.

Request_Type Statements

IMFT requires the following Request_type statement and substatements in the I/O daemon tables. If you wish to use a queue other than the default queue, use the max_queues statement (see the Bulk I/O manual).

Request_type: <name>;
: Defines the name of the Request_type and denotes the beginning of a Request_type description. Any subsequent statements (see example) apply to this Request_type until the next Line, Request_type, or Device statement is encountered. <name> must be the <name> specified as the foreign system name in the input and output driver definitions, prefaced by the string "From_" or "To_" (e.g., To_MIT).

generic_type: <name>;
: IMFT requires the generic type <name> to be: imft.

driver_userid: <Person_id.Project_id>;
: Must identify the user selected above to run this connection.

default_queue: <number>;
: Specifies the default queue, where <number> is the default queue number. <number> must be between 1 and 4.

accounting: <name>;
: IMFT requires <name> to be "nothing".

max_access_class: system_high;
: Must be specified exactly as shown, or the I/O coordinator will leave requests in the queues indefinitely.

device: <name>;

Specifies the devices that can be used to process requests of the associated type. If you have a minor output device that is for submitting transfers to the remote site, specify it in the device statement of the corresponding Request_type (e.g., the one that begins with the string "To_"). Similarly, if you have a minor device that is for requesting file transfers from the remote site, specify it in the device statement of the corresponding Request_type (e.g., the one that begins with the string "From_"). Specify the input driver in the device statement of **either** the Request_type for transfers to the foreign site, or the Request_type for transfers from the foreign site. (If you have only defined one minor output device, give the input driver the same Request_type as that of the output minor device.)

Example

In the example below, a pair of Request_types are defined for an IMFT connection with the system named "System-M". The first Request_type specifies the input driver and the minor output device used for transfers to the remote site. The second Request_type specifies the minor output device used for transfers from the remote site.

```
Request_type:                   To_System-M;
  generic_type:                 imft;
  driver_userid:                IMFT.Daemon;
  default_queue:                3;
  accounting:                   nothing;
  max_access_class:             system_high;
  device:                       system_m_ft_out.to;
  device:                       system_m_ft_in;

Request_type:                   From_System-M;
  generic_type:                 imft;
  driver_userid:                IMFT.Daemon;
  default_queue:                3;
  accounting:                   nothing;
  max_access_class:             system_high;
  device:                       system_m_ft_out.from;
```

IMFT Project Master File Entries

The list of persons who can log in on a project is contained in a binary table known as the Project Definition Table (PDT), maintained by the system. There is one entry in the PDT for each user; the entry contains the user's attributes and resource usage information. The PDT is created from an ASCII segment called the Project Master File (PMF). A PMF exists for each registered project. It is the basic project-administration data base and contains the project's specification of user attributes. A PMF entry must be defined for the IMFT daemon process.

The system references the PDT, not the PMF, when it determines if a user can log in. To make a change to the PDT, modify the PMF, convert the PMF into a binary copy of the PDT via the cv_pmf command, and request the system to modify its PDT according to the copy using the install command (see MAM Project manual).

Every project on the system has a "Project_id" in the PMF. A separate Project_id can be created for IMFT. If IMFT is to be available on a system wide basis, add the required PMF entries to a system overhead project such as "Daemon" or "SysDaemon". If IMFT is to be available for a small group of users in a specific project, add the required entries to that project entry in the PMF.

The following PMF entries are required to define the process that will run IMFT:

Project_id: character_string;
: The name of the Project_id under which IMFT will run. If you are not establishing a separate project for IMFT, do not enter a Project_id and place the keywords below in a system overhead project such as "Daemon", or a user project.

Person_id: character_string;
: character_string is the Multics Person_id. It is recommended that you use "IMFT" as the Person_id.

initproc: pathname;
: For IMFT, pathname must be "iod_overseer_".

attributes: character_strings;
: Character_strings are attribute names separated by commas. IMFT requires the following attribute names (for additional entries, see the MAM Project manual):

^v_process_overseer	Indicates that no other process overseer can be specified for IMFT.
dialok	IMFT can accept dial requests.
daemon	IMFT can be logged in by the operator via the message coordinator.
multip	Permits multiple logins for IMFT: one for the input driver and one for the output driver.

Example

```
Person_id:  IMFT;
  initproc:  iod_overseer_;
  attributes:  ^v_process_overseer, dialok, daemon, multip;
```

In the above example, the required PMF entries for IMFT are placed under an existing Project_id.

IMFT Person Name Table Entries

The IMFT daemon, local system name, and foreign system name must be registered, as 3 individual users would, in the Person Name Table (PNT) and the User Registration File (URF). To register IMFT in both, use the system administrator "new_user$nua" command (see the MAM System manual).

The driver processes of an IMFT connection perform an initial "handshake" sequence. During the handshake sequence, each driver transmits the Person_id and card input password specified in the local_system keyword in the args statement (see "Device Definition for IMFT"). The remote system validates the Person_id and password against the Person_id and password specified by the foreign_system keyword. Therefore, register the Person_ids specified in the local_system keyword and the foreign_system keyword on both systems with the same card input passwords.

When registering the daemon, preface the full name with an asterisk (*) to allow the system to prompt for the Person_id. For default project, specify the Project_id specified in the PMF; no password is required but one can be specified; and no card reader password is required. For AIM authorizations, enter the entries decided upon from "AIM Considerations".

Local system registration requires that you preface the full name with an asterisk (*) to allow the system to prompt for a Person_id; no password is required. The card reader password specified must be the card reader password given on the remote system when this Person_id was registered.

Preface the full name of the foreign system with an asterisk (*) to allow the system to prompt for a Person_id. No password is required. The card password must be the same as the one given on the remote system when this Person_id was registered.

PREPARING IMFT FOR OPERATION

To run an IMFT connection, you must:

- Assign the appropriate access to the PNT
- Create an access control segment (ACS) for the subchannels (HASP or X.25) and assign the appropriate access to the ACS

- Assign the appropriate access to the system_privilege_ gate
- Prepare the message coordinator for IMFT
- Optionally, add entries to your system_start_up.ec or admin.ec to simplify operation of the drivers.

Assigning Access to the PNT

To validate the remote drivers while establishing a connection, and also to validate users during the actual transfer of requests, give the processes that run the input and output drivers at least read (r) access to the system's Person Name Table (PNT).

To assign the correct access, issue a set_access command as in the following example:

```
set_access >sc1>PNT r Person_id.Project_id
```

Subchannel Access Control Segment

To create the required ACS and set the appropriate access for the subchannels of either a HASP or an X.25 multiplexed channel, issue the following command:

```
create >sc1>rcp>x.hNNN.sub1.acs
```

where x.hNNN is the channel number specified in the CMF, and sub1 identifies one of the multiplexer subchannels identified in the args statement of the input and output drivers.

Next, issue an add_names command employing the equals convention to associate the rest of the subchannels with the newly created ACS, as in the following example.

```
add_names >sc1>rcp>x.hNNN.sub1.acs =.=.sub2.= =.=.sub3.=
=.=.sub4.=
```

where sub2, sub3, and sub4 identify the rest of the subchannels in the args statement.

Finally, set the appropriate access to the ACS as in the following example.

```
set_access >sc1>rcp>x.hNNN.sub1.acs rw Person_id.Project_id
```

system_privilege_ Access

The IMFT driver requires access to the system_privilege_ gate. Therefore, add the following command line to the system_start_up.ec for each user who will run an IMFT connection:

```
hp_set_acl >sl1>system_privilege_ re Person_id.Project_id
```

queue_admin_ Access

An input driver that processes requests for remote transfer (i.e., "allow_remote_request=yes" appears in the device definition) requires access to the queue_admin_ gate. Therefore, add the following command line to the system_start_up.ec for each user who will run such an input driver:

```
hp_set_acl >sl1>queue_admin_ re Person_id.Project_id
```

Message Coordinator

For the IMFT daemon to issue messages and receive directives, you must create the input and output message segments, and establish the message coordinator sources and virtual consoles.

Issue the following command only once to create the input and output message segments:

```
create >sc1>(input_SOURCE_name output_SOURCE_name).message
```

Use the define command and the route command to establish the message coordinator sources and virtual consoles. Most often the two commands are entered in the system_start_up.ec file.

IMFT messages can be sent to an actual terminal, a log file, or both an actual terminal and a log file. If you want to send IMFT messages to a terminal only, or a terminal and a log file, you must register a message coordinator channel for the terminal in the CMF (see MAM Communications) and issue an accept command (see MOH) in addition to the define command and route command.

Following is a brief description of the define and route commands; for a full description, see the MOH.

DEFINE COMMAND

The define command defines the virtual console that will receive messages from the IMFT daemon. The format of the define command is:

define VCONS TYPE DEST

Example 1

```
sc_command define iod tty b.h200
```

In the above example, a virtual console named iod is defined that forwards all output sent to it to the terminal whose channel is b.h200

Example 2

```
sc_command define iolog log iolog
```

In the above example, a virtual console named iolog is defined that forwards all output sent to it to the log file named >sci>iolog.

ROUTE COMMAND

The route command sends output from the IMFT daemon to the designated virtual console. A route command must be issued for the user_i/o, error_i/o, and log_i/o streams of both the input and output drivers. Thus, a total of six route commands must be issued for each virtual console that is to receive IMFT output. The format of the route command is:

route SOURCE STREAM VCONS

Example

```
sc_command route (mitfti mitfto) user_i/o iod
sc_command route (mitfti mitfto) error_i/o *iod
sc_command route (mitfti mitfto) log_i/o iod
sc_command route (mitfti mitfto) user_i/o iolog
sc_command route (mitfti mitfto) error_i/o *iolog
sc_command route (mitfti mitfto) log_i/o iolog
```

In the above example, output from the IMFT daemons using the sources mitfti and mitfto is routed to the two virtual consoles iod (a terminal) and iolog (a log file) defined in the previous examples. The asterisk (*) before the virtual console name for the error_i/o stream causes the terminal to issue an audible alarm (beep tone) whenever error messages are issued. Command iteration is used to reduce the number of route commands lines from twelve to six. Note that the commands are prefaced with the sc_command as required when adding operator commands to an ec file.

IMFT Initialization

You can configure IMFT to run automatically without operator intervention, to run with other daemon processes that are invoked by the operator, or to run separately from other daemon processes.

If you configure IMFT to run automatically without operator intervention (see "Device Type Definition for IMFT"), place the login commands that log in the IMFT input and output driver processes and the reply commands (below) in the system_start_up.ec file.

If you decide to initialize IMFT with other daemon processes, edit the login commands and reply commands into the admin.ec segment that contains the definitions of the daemon processes invoked by an operator x command.

If you are not running IMFT with automatic initialization, or are not initializing IMFT with other daemon processes, you can create a new entry in admin.ec that is invoked with a unique operator x command. The load_mpx command (see the MOH) might then be then required with replies of receive and go.

LOGIN COMMAND

The login command causes the login of a daemon process at the operator's request. For a full description of the login command, see the MOH.

The format of the login command is:

login user_id SOURCE

It is recommended that you issue a pause command directly after the login command in the system_start_up.ec file, as follows:

pause 10

This gives the system time to log in the daemon process before executing subsequent commands in the file.

Example

```
sc_command login IMFT.Daemon mitfti
sc_command login IMFT.Daemon mitfto
```

In the above example, the IMFT input and output processes are logged in.

REPLY COMMAND

The reply command sends an input line to a specified source and sends a wakeup to that source.

An IMFT driver requires at least two lines of input before it can begin operation. The first line is "driver"; the second line is one of the two major device names given in the I/O daemon tables for the connection. For an output driver, the major device name should be followed by the word "default", so that the minor devices can be initialized without further operator intervention.

If the driver is operating in manual mode, a third line, either "go" or "receive", must be sent once the driver announces that it is ready after the operator establishes the physical connection.

The format of the reply command is:

reply SOURCE REST_OF_LINE

where SOURCE is the source to which the input line is to be sent .and REST_OF_LINE is the input line to be sent.

Example

```
sc_command reply (mitfti mitfto) driver
sc_command reply mitfti mit_file_transfer_in
sc_command reply mitfto mit_file_transfer_out default
```

SECTION 3

OPERATOR PROCEDURES

This section describes the normal operation of an IMFT connection with a remote Multics system. The instructions given in this section should be performed on both the local and foreign systems to ensure proper system operation.

Operating procedures can vary depending on how IMFT has been installed on your system. Check with your system administrator regarding initialization procedures for IMFT specific to your site.

INITIALIZING THE IMFT CONNECTION

To initialize the IMFT connection:

1. Login the two driver processes for the IMFT connection by either issuing the appropriate operator x command if the login commands are contained in an ec file, or issuing the sequence used to login a driver process described in "Login and Initialization of Device Drivers" in the Bulk I/O manual. In brief, the commands given to each process are:

   driver
   <device_name> {default}

   where <device_name> is one of the two device names for the drivers specified in the I/O daemon tables. (For example, the <device_name>s from two of the examples given in Section 2 of this manual are "system_m_ft_out" and "mit_file_transfer_in"). If a driver has one or more minor devices (e.g., an output driver), the major device name should be followed by the word "default".

2. Establish the physical connection with the remote system. If a hardwired connection is used, the connection should be established automatically after both systems are running. For dial-up connections, it may be necessary to issue the "load_mpx" command as described in the MOH before making or receiving the phone call to complete the connection. Check with your system administrators for details on establishing the connection at your site.

3. After the drivers indicate that they are ready, it may be necessary to issue a "receive" command to the input driver and a "go" command to the output driver. Again, check with your system administrators to determine if these commands are necessary.

RUNNING THE IMFT DRIVER PROCESS

Some of the commands for an IMFT output driver process are different from those used on an IMFT input driver process. To avoid problems, we recommend that you use only the commands suggested below for each driver.

The IMFT Input Driver

The only I/O daemon commands that you should use on a regular basis in an IMFT input driver process are: logout (described in Bulk I/O) and receive (described below). You may occasionally be requested by the system administrator to issue one of the following commands: hold, start, reinit (described in Bulk I/O). However, unless you receive a specific request to do otherwise, use only logout and receive on the input driver.

The output driver process is the correct place from which to cancel or defer a running request. Therefore, if you are operating an input driver and you are asked to cancel/defer a request, contact the operator of the appropriate output driver and ask him to cancel/defer the request.

The IMFT Output Driver

In addition to the commands used to defer or cancel a running request, the following commands can be used when operating an output driver: go, hold, logout, reinit, start, and status. These commands and all of the commands mentioned below are fully described in the Bulk I/O manual.

CANCELLING A RUNNING REQUEST

A running request is cancelled from the output driver. To cancel a running request:

1. Press the <QUIT> or <BREAK> key.

2. After the system prompts you, type the "cancel" command.

The daemon prints a message to the effect that the request has been cancelled, and then proceeds to the next request. Once the initial request has been cancelled, the user must resubmit the request in order to run it again.

DEFERRING A RUNNING REQUEST

You must defer a running request from the output driver site. The two deferral methods are described below. Keep in mind that after a request is deferred, when the driver runs it again it starts from the beginning of the file. If the request you are deferring is a very long one, you may want to use the "defer_time" command and run it again at a much later, more convenient time.

When you defer a running request, you must tell the driver which request you want it to process next. By default, the driver begins processing the next request in the queue. If this is what you want it to do, type the following commands:

1. Press the <QUIT> or <BREAK> key.

2. After the prompt, type the "defer" command. This causes the driver to defer the current request, and proceed to the next request in the queue.

If you want to defer the running request and tell the driver to process a specific request (i.e., NOT the next request in the queue), type the following set of commands:

1. Press the <QUIT> or <BREAK> key.

2. After the prompt, type the "hold" command. This causes the daemon to stay at command level until you type "go".

3. After the prompt, type the "defer" command. This command sends the current request back to its queue in its original position, marked as "deferred".

4. After the prompt, type the "next" command, specifying the following arguments: -user Person_id, -device to/from, and either -entry STR, -path path, or -id ID. This command specifies which request is to be run next by the driver. If the request is for a file to be sent to the foreign site, specify "-device to"; if it is for a file to be sent from the foreign site, specify "-device from". Note that you must supply one of the following three request identifiers: the entryname, the pathname, or the id number of the request. For example, the operator types this command line to request that user Ferron's request for the file "data.pl1", which is to be sent to the foreign site, is run next.

   next -device to -user Ferron -entry data.pl1

5. This step is optional. If you want the deferred request to run again right after the request specified in the "next" command, type either "restart_q to" or "restart_q from", depending on whether the deferred request is at the local site (to) or at the foreign site (from).

6. If the system tells you the request was not found, check the "next" command line, and enter the command again. If the system tells you the request was found, type the "go" command in response to the prompt. The driver will begin processing the specified request. When that request is completed, the driver will select the next queue entry with the highest priority, and continue processing as usual.

TERMINATING AN IMFT CONNECTION

To terminate operation of an IMFT connection:

1. Issue the "logout" command to each of the two driver processes for the connection.

2. After the driver processes have logged out, terminate the physical connection between the two systems by hanging up the phone. If a hardwired connection is used, no further action need be taken. For dialup connections, it may be necessary to issue the "dump_mpx" command as described described in the MOH before hanging up the phone. Check with your system administrators for details on breaking the physical connection at your site.

Name: receive

The receive command causes an IMFT input driver to wait for files or subtrees to be transmitted from the remote Multics system. Messages are issued at the start and end of each file or subtree received.

Usage

receive

Notes

If the "auto_receive=yes" parameter is specified in the I/O daemon tables for an input driver, a receive command is automatically issued when the driver becomes ready. In this case, the driver is capable of operation without any operator intervention.

SECTION 4

USER COMMANDS

The Inter-Multics File Transfer Facility (IMFT) allows files and subtrees to be transferred between Multics systems. IMFT is queue driven, meaning that your requests are placed in a queue for later action. IMFT supports the following user commands:

- enter_imft_request (eir)
  submits an IMFT request to transfer a file to or from a site
- list_imft_requests (lir)
  lists the IMFT requests in the specified queue at the source site or the target site
- cancel_imft_request (cir)
  cancels an IMFT request from a specified queue at the source site or the target site
- move_imft_request (mir)
  moves an IMFT request from one priority queue to another at the source site or the target site
- print_imft_sites
  displays the names of foreign sites that can be used with the -source or -destination control arguments of the enter_imft_request command.

You can request IMFT to transfer files from the system where you are logged in (the "local" system), to some other system (the "remote" or "foreign" system). You can also request IMFT to transfer files from the remote system to the local system. In the following discussion, the system from which the files are transferred is called the source system, and the system to which they are being transferred is called the target system. (Note that the local system can be either the source or the target system, depending on whether it is receiving or sending the file; similarly, the foreign system can be either the source or the target system.)

ACCESS REQUIREMENTS

To transfer a file or a subtree from the source system to the target system, the conditions detailed below must be met.

For files, the user on the source system must have at least "r" access to the file; for subtrees, the user must have at least "s" access to the root of the subtree and each directory contained therein and at least "r" access to each file in the subtree.

The daemon process on the source system that transfers the file or subtree must also have the same type of access as described above for the source system's user. Additionally, the daemon must also have at least "s" access to the directory containing the file or subtree in order to verify that the user has the proper access. The identity of the daemon can be determined using the print_imft_sites command.

The user on the target system must have "sma" access to the directory into which the file or subtree will be placed. The source system user and the target system user are the same unless the -foreign_user control argument is specified.

The daemon process on the target system that receives the file or subtree must also have "sma" access to the directory into which the file or subtree will be placed. In addition, this daemon must have at least "s" access to the directory containing that directory in order to validate that the target user has the proper access.

In order for the user on the local system (LPerson.LProj) to be able to transfer files to or from the foreign system, the user on the foreign system (FPerson.FProj) must give her access to the segment:

>udd>FProj>FPerson>LSite.imft.acs

on the foreign system where LSite is the name of the local system. If she wants files to be transferred from the foreign system, LPerson.LProj must have read access to the above-named segment. To transfer files to the foreign system, she needs write access to the segment. (Note: when setting write access on an ACS, you should set its maximum length to 0 to prevent it from acquiring contents. See the set_max_length command in the Commands manual for details.)

A remote request is one which makes use of the -source control argument to transfer a file from the foreign system. The site administrator may choose to restrict transfer of files by remote request to those files whose ACLs have explicit terms for the IMFT daemon. In this case, an ACL term of "r *.*.*" is not sufficient to permit the file to be transferred.

To determine the identity of the daemon on the foreign system and the name of the local system used to form the name of the ACS segment above, use the print_imft_sites command on the foreign system.

Regarding ring brackets, IMFT insures that the ring brackets of all segments and directories it creates on the target system are never less than the write bracket on the ACS segment of the user receiving the file or subtree. This eliminates potential problems if, for example, the user operates on the source system in ring-4 and on the target system in ring-5. When that user issues a transfer request for a segment, IMFT will create the segment in ring-5 on the target system.

Assume that user Bell.SysMaint on MIT wishes to send the file:

```
>udd>sm>pbk>test>new_version.pl1
```

to the directory:

```
>udd>ssa>pbk>imft>mit
```

on System-M where his User_ID is PBell.SiteSA. Further, assume that the daemon on both systems is IMFT.Daemon, and the names of the source and target systems (given by the print_imft_sites command) are MIT and System-M respectively.

On MIT (the source system), Bell.SysMaint issues the following set_acl commands to ensure that he and the daemon have proper access:

```
set_acl >udd>sm>pbk>test>new_version.pl1 r Bell.* r IMFT.Daemon
set_acl >udd>sm>pbk>test s IMFT.*
```

Note that here, any ACL term which grants appropriate access is sufficient. In other words, an ACL term on >udd>sm>pbk>test for IMFT.Daemon.*, IMFT.*.*, *.Multics.*, or even *.*.* is sufficient to give the daemon proper access; it is not necessary to use an ACL term for IMFT.Daemon.* explicitly, although that is acceptable.

On System-M, PBell.SiteSA issues the following set_acl commands to ensure proper access to receive the file:

```
set_acl >udd>ssa>pbk>imft>mit sma PBell.* sma IMFT.Daemon
set_acl >udd>ssa>pbk>imft s IMFT.Daemon
set_acl >udd>SiteSA>PBell>MIT.imft.acs w Bell.SysMaint
```

Once proper access is established, Bell.SysMaint issues the command line:

```
eir >udd>sm>pbk>test>new_version.pl1 -tpn
 >udd>ssa>pbk>imft>mit>=== -fu PBell.SiteSA -ds System-M
```

Let's assume that the same user wants to transfer the same file as a remote request (i.e., he wants to issue the request while logged in at System-M as PBell.SiteSA.) To do this, he needs exactly the same access as described above, with one exception. Instead of giving himself "w" access to the ACS segment, he must establish "r" access with the following command line at MIT:

```
set_acl >udd>Sys_Maint>Bell>System-M.imft.acs r PBell.SiteSA
```

PBell.SiteSA can now request the file transfer with the command line:

```
eir >udd>sm>pbk>test>new_version.pl1 -tpn
 >udd>ssa>pbk>imft>mit>=== -fu Bell.SysMaint -source MIT
```

Notes on AIM

When using AIM, files and subtrees can be transferred only if their access class is less than or equal to (1) the process authorization of the user who submits the request and (2) the common access class ceiling.

COMMON ACCESS CLASS CEILING

The common access class ceiling between two systems is determined by locating the overlapping AIM attributes within the sensitivity levels and access categories specified for the two systems.

The common access class ceiling is defined as:

- All sensitivity levels from level 0 (usually un-named) up to but not including the first level that does not have the same long and short name on both systems, and
- All access categories that have the same long and short names on both systems.

If the long and short names of sensitivity level 0 are not the same on both systems, then the two systems have no common access ceiling and are isolated from each other.

For example, if system A defines the following AIM attributes:

```
level 0          *-* UN-NAMED *-*
level 1          unclassified              u
level 2          secret                    s
level 3          top secret                ts

category 1       SSTD                      sstd
category 2       LISD                      lisd
category 3       FSD                       (none)
category 4       Marketing                 (none)
```

and system B defines the following attributes:

```
level 0          *-* UN-NAMED *-*
level 1          unclassified            u
level 2          restricted              (none)

category 1       MPO                     (none)
category 2       LISD                    lisd
category 3       FSD                     fsd
category 4       SSTD                    sstd
```

then the common access ceiling is:

unclassified, LISD, SSTD

The sites may choose to lower the common access class ceiling, so check with your system administrator to find out the actual common access class ceiling.

Files and subtrees are created on the foreign system with the same access class that they had on the local system.

When transferring a subtree, the daemon will not transfer any directory in the subtree if its access class is greater than that of the directory where it will be placed on the foreign system. Therefore, it is necessary to issue separate requests for each such upgraded directory, specifying a target pathname whose containing directory has the same access class as the directory being transferred.

For example, if the access class of the directory on the local system is "classified, LISD" and the command line:

```
eir my_directory -tpn >udd>m>ghm>receiver>===
```

is issued, the access class of the directory >udd>m>ghm>receiver on the foreign system must also be "classified, LISD".

USER COMMANDS

With the following user commands you can enter, list, cancel, and move IMFT requests, as well as display the names of IMFT sites.

enter_imft_request eir

Name: enter_imft_request eir

The enter_imft_request command submits requests to transfer files or subtrees to or from remote Multics systems using the Inter-Multics File Transfer (IMFT) facility.

Usage

eir transfer_specs {-control_args}

where:

1. transfer_specs
   specify the files or subtree to be transferred and has the following format:

   path {-target_pathname equal_path},
   path {-tpn equal_path}
   path specifies the relative pathname of files and/or subtrees to be transferred. The star convention is accepted. If supplied, the equal_path is the relative pathname of where the files and subtrees will be placed on the target system. The equal convention is accepted. The target pathname is converted to an absolute pathname relative to the working directory on the source system. If not given, the files and subtrees are given the same pathname on the target system.

2. control_args
   may be chosen from the following:

   -absolute_pathname, -absp
   prints the absolute pathname of the file or subtree along with the request ID for each request entered by this command.

   -brief, -bf
   suppresses the messages providing the request IDs of the requests entered by this command.

   -chase
   specifies that transfer requests are issued for the targets of any links which match the transfer_specs. The default is to a) chase links for any transfer_specs that do not use the star convention, and b) do not chase links for any transfer_specs that use the star convention.

   -destination STR, -ds STR
   identifies the foreign system to which the files and subtrees are to be transferred. STR must be one of the names listed by the print_imft_sites command. The default STR is imft. If neither -destination nor -source is specified, the default is -destination.

enter_imft_request eir

-foreign_user Person.Project, -fu Person.Project
specifies the identity of the user at the foreign system for whom the transfer requests are being entered. Notifications on the foreign system are sent to this user. See "Access required" below for further information. The default is that the foreign user is the same as the local user.

-entryname, -etnm
prints only the entry name of the file or subtree along with the request ID for each request entered by this command. This is the default.

-file, -f
specifies that transfer requests are issued only for files which match the transfer_specs. If a transfer_spec does not use the star convention and there is no matching file, an error message is issued. The default is to issue requests for matching files.

-long, -lg
prints the messages providing the request IDs of the requests entered by this command. This is the default.

-long_id, -lgid
prints the long form of the request ID in any messages.

-merge_directories, -mdr
specifies that if there is a directory on the target system with the same name as one of the names on the root directory of the subtree being transferred, the contents of the source subtree are merged with the target subtree. If the target entry is not a directory, processing continues as though -replace_directories had been specified. Any directories within the subtree are treated in a similar fashion with respect to name duplications. See "Notes" for a description of the treatment of files within the subtree. This is the default.

-notify, -nt
sends notification of successful initiation and completion of each transfer request. The notifications are sent on the source and target systems. This is the default.

-no_chase
specifies that transfer requests are not issued for the targets of any links which match the transfer_specs.

-no_notify, -nnt
suppresses notification of successful transfer on both systems. Any errors detected during transmission will still generate mail, regardless of the use of -no_notify.

enter_imft_request eir

-queue N, -q N
specifies that the requests are entered in priority queue N, where N is an integer between 1 and 4 inclusive. The default depends on the destination or source specified.

-replace_directories, -rpdr
specifies that if there is an entry on the target system with the same name as one of the names on the root directory of the subtree being transferred, that name is removed from the target entry; if the target entry has only one name, it is deleted.

-short_id, -shid
prints the short form of the request ID. This is the default.

-source STR, -sc STR
identifies the foreign system from which the files and subtrees are to be transferred. STR must be one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

-subtree, -subt
specifies that transfer requests are issued only for subtrees which match the transfer_specs. If a transfer_spec does not use the star convention and there is no matching subtree, an error message is issued. The default is to issue requests for matching subtrees.

Notes

If conflicting control arguments (e.g., -notify and -no_notify, or -destination and -source) are given on the command line, the rightmost control argument takes effect.

If there is an entry on the target system with the same name as one of the names on the file being transferred, that name is removed from the target entry; if the target entry has only one name, it is deleted. No distinction is made here between files specified in a transfer_spec and files contained in a subtree.

Examples

eir **.pl1 -tpn <x>===.new -ds MIT
transfers all files and subtrees in the working directory whose names end with the pl1 suffix. If the source working directory is >udd>m>gmp>w, a file named "foo.pl1" appears on the target system as ">udd>m>gmp>x>foo.pl1.new".

enter_imft_request eir

eir my_subtree -ds System-M -mdr
transfers the subtree named "my_subtree" in the working directory to the same point in the hierarchy on the target system. Assume (1) that there is already a target directory named my_subtree, (2) that the source my_subtree contains two files named file1 and file2 and a directory named subdir1, and (3) that the target my_subtree also contains two files named file1 and file3. After the transfer is completed, the target my_subtree contains three files (file1 and file2 from the source system and file3 from the target system) and one directory (subdir1 from the source system), along with the contents of the source subdir1.

eir >udd>sm>Brown.profile -tpn >udd>m>PBrown.= -source MIT
-fu Brown.SysMaint

from the local system, requests a transfer of the segment >udd>sm>Brown.profile from MIT on behalf of MIT user Brown.SysMaint, to be placed in the file >udd>m>PBrown.profile on the local system.

list_imft_requests lir

Name: list_imft_requests lir

The list_imft_requests command lists requests in the Inter-Multics File Transfer queues.

Usage

lir {request_identifiers} {-control_args}

where:

1. request_identifiers
   determine which requests in the selected queues belonging to the specified users are listed. If not given, all of the appropriate requests are listed. See "Notes on request identifiers" below.

   List of request_identifiers:

   path
   lists all requests from the appropriate queues and users whose source pathnames match the relative pathname path. The star convention is allowed.

   -entry STR, -et STR
   lists all requests from the appropriate queues and users whose source entry names match STR; the directory portions of the source pathnames are ignored. The star convention is allowed.

   -id STR
   lists all requests from the appropriate queues and users whose request IDs match the STR. Type "help request_ids.gi" for a description of the syntax of STR.

2. control_args
   may be chosen from the following:

   -absolute_pathname, -absp
   displays the absolute pathname of the file or subtree associated with each request. This is the default if -long is used.

   -admin, -am
   lists the matching requests submitted by any user.

   -all, -a
   lists requests entered in all priority queues for the above destination.

   -brief, -bf
   displays minimal information for each request including its request ID, source pathname, and current state. This is the default.

list_imft_requests lir

-destination STR, -ds STR
lists requests that are queued for transfer to the foreign system identified by STR. STR must be one of the names listed by the print_imft_sites command. (The default STR is imft.) If neither -destination nor -source is specified, the default is -destination.

-entryname, -etnm
displays only the entry name of the file or subtree. This is the default if -long is not used.

-long, -lg
displays all information available for each request.

-long_id, -lgid
displays the complete request ID for each request. This is the default if -long is used.

-no_position, -npsn
does not display the queue position of each request. This is the default.

-own
lists a matching request only if it was submitted by the user of this command. This is the default.

-position, -psn
displays the position within the queue of each request.

-queue N, -q N
lists requests entered in priority queue N, where N is an integer between 1 and 4 inclusive. The default depends on the destination or source specified.

-short_id, -shid
displays the short form of the request ID for each request. This is the default if -long is not used.

-source STR, -sc STR
identifies the foreign system from which the files and subtrees are to be transferred. STR must one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

-total, -tt
displays only the total number of matching requests in each queue.

list_imft_requests lir

-user STR
lists a matching request only if it was submitted by the user identified by STR. STR must have one of the following forms:

Person.Project
lists only those matching requests entered by the specified user while logged in on the specified project.

Person.*, Person
lists only those matching requests entered by the specified user while logged in on any project.

*.Project, .Project
lists only those matching requests entered by any user logged in on the specified project.

., *
lists all matching requests regardless of who entered them.

Access required: If -position, -admin, or -user is specified, at least "r" extended access is required to the queues; otherwise, at least "o" extended access is required.

Notes on request identifiers: If path or -entry STR request identifiers are given, only one -id STR request identifier may be given. Further, only those requests which match one of the path or -entry STR identifiers and which match the -id STR identifier are listed.

Notes

If conflicting control arguments (e.g., -position, -no_position) are given on the command line, the rightmost control argument takes effect.

Examples

Assume that you have "r" extended access to the queues, and that the following files are in imft queue 2. Further, assume that queue 2 is the default queue.

```
User                    ID         Entry name
Carey.System-M          140628.3   mtape.pl1
Poole.ProjA             163146.3   mcr.packet.j31
Randolph.System-M       160200.7   jm_write.pl1
```

list_imft_requests lir

User Poole.ProjA enters the following command, which lists each request submitted by him:

```
lir
```

This command line gives the following results:

```
imft queue 2:       1 request; 3 total requests.

  163146.3   mcr.packet.j31
```

The following command line lists the requests submitted by any user, and specifies the queue position of each request.

```
lir -admin
```

This command line evokes the following response:

```
imft queue 2:      3 requests; 3 total requests.

User                          ID        Entry name
Carey.System-M          1)  140628.3  mtape.pl1
Poole.ProjA             2)  163146.3  mcr.packet.j31
Randolph.System-M       3)  160200.7  jm_write.pl1
```

The following command line lists any request whose entry name ends with the pl1 suffix, regardless of who made the request:

```
lir -user *.* -et *.pl1
```

This command line generates the following response:

```
imft queue 2:      2 requests; 3 total requests.

User                          ID        Entry name
Carey.System-M              140628.3  mtape.pl1
Randolph.System-M           160200.7  jm_write.pl1
```

cancel_imft_request cir

Name: cancel_imft_request cir

The cancel_imft_request command cancels requests in the Inter-Multics File Transfer queues.

Usage

cir request_identifiers {-control_args}

where:

1. request_identifiers
   determine which requests in the selected queues belonging to the specified users are cancelled. See "Notes on request identifiers" below.

   List of request_identifiers:

   path
   cancels all requests from the appropriate queues and users whose local pathnames match the relative pathname path. The star convention is allowed.

   -entry STR, -et STR
   cancels all requests from the appropriate queues and users whose local entry names match STR; the directory portions of the local pathnames are ignored. The star convention is allowed.

   -id STR
   cancels all requests from the appropriate queues and users whose request IDs match the STR. Type "help request_ids.gi" for a description of the syntax of STR.

2. control_args
   may be chosen from the following:

   -all, -a
   cancels requests entered in all priority queues for the above destination.

   -destination STR, -ds STR
   cancels requests that are queued for transfer to the remote system identified by STR. (The default STR is imft.) STR must be one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

cancel_imft_request cir

-own
cancels a matching request only if it was submitted by the user of this command. This is the default.

-queue N, -q N
cancels requests entered in priority queue N, where N is an integer between 1 and 4 inclusive. The default depends on the source or destination specified.

-source STR, -sc STR
cancels requests for transferring files and/or subtrees from the remote system identified by STR. STR must one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

-user STR
cancels a matching request only if it was submitted by the user identified by STR. STR must have one of the following forms:

Person.Project
cancels only those matching requests entered by the specified user while logged in on the specified project.

Person.*, Person
cancels only those matching requests entered by the specified user while logged in on any project.

*.Project, .Project
cancels only those matching requests entered by any user logged in on the specified project.

., *
cancels all matching requests regardless of who entered them.

Access required: If -user is specified, at least "rd" extended access is required to the queues; otherwise, at least "o" extended access is required.

Notes on request identifiers: If path or -entry STR request identifiers are given, only one -id STR request identifier may be given. In this case only those requests which match one of the path or -entry STR identifiers and which match the -id STR identifier are listed.

Notes

If conflicting control arguments (e.g., -queue 2, -queue 3) are given on the command line, the rightmost control argument takes effect.

cancel_imft_request cir

Examples

Assume that you have "o" access to the imft queues, and that you have transfer requests for the following files in queue 2:

```
s5.mrg.compin
s7.mrg.compin
stm.pl1
```

and a transfer request for P226.packet in queue 3.

The following command cancels a request in any queue for a file ending with the suffix "packet".

```
cir *.packet -all
```

It evokes this response:

```
IMFT request P226.packet cancelled from queue 3.
r 10:03 0.231 2
```

The following command cancels each file in the default queue that has an entry name ending in the suffix compin:

```
cir -et **.compin
```

The system gives you the following response:

```
IMFT request s5.mrg.compin cancelled.
IMFT request s7.mrg.compin cancelled.
```

move_imft_request mir

Name: move_imft_request mir

The move_imft_request command moves requests from one Inter-Multics File Transfer (IMFT) queue to another. The move can be between queues of the same remote system, or between queues of different remote systems. When a request is moved, it is always placed at the end of the "new" queue.

Usage

mir request_identifiers -control_args

where:

1. request_identifiers
   determine which requests in the selected queues belonging to the specified users are moved. See "Notes on request identifiers" below.

   List of request_identifiers:

   path
   moves all requests from the appropriate queues and users whose source pathnames match the relative pathname path. The star convention is allowed.

   -entry STR, -et STR
   moves all requests from the appropriate queues and users whose source entry names match STR; the directory portions of the source pathnames are ignored. The star convention is allowed.

   -id STR
   moves all requests from the appropriate queues and users whose request IDs match the STR. Type "help request_ids.gi" for a description of the syntax of STR.

2. control_args
   may be chosen from the following:

   -destination STR, -ds STR
   moves requests that are queued for transfer to the foreign system identified by STR. (The default STR is imft.) STR must be one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

move_imft_request mir

-source STR, -sc STR
identifies the foreign system from which the files and subtrees are to be transferred. STR must one of the names listed by the print_imft_sites command. If neither -destination nor -source is specified, the default is -destination.

-queue N, -q N
moves requests entered in priority queue N for the above destination or source where N is an integer between 1 and 4 inclusive. The default depends on the destination specified.

-all, -a
moves requests entered in all priority queues for the above destination or source. If the move is between queues of the same foreign system, the target queue is not searched.

-brief, -bf
suppresses messages that either tell that a particular request identifier did not match any requests, or provide the new request ID of the moved request.

-long, -lg
displays the above messages.

-to_destination STR, -tods STR
specifies that the requests are moved to the queues of the target system identified by STR. The default is that requests are moved within the queues of the target system given by the -ds STR control argument.

-to_source STR, -tosc STR
specifies that the requests should be moved to the input queues of the remote system identified by STR. This control argument cannot be specified unless -source is also specified. By default, requests are moved within the queues of the remote system given by the -sc STR control argument.

-to_queue N, -tq N
specifies that the requests are moved to priority queue N of the target or source system, where N is an integer between 1 and 4 inclusive. The default value of N is the default queue of the target or source system.

-own
moves a matching request only if it was submitted by the user of this command. This is the default.

move_imft_request mir

-user STR
moves a matching request only if it was submitted by the user identified by STR. STR must have one of the following forms:

Person.Project
moves only those matching requests entered by the specified user while logged in on the specified project.

Person.*, Person
moves only those matching requests entered by the specified user while logged in on any project.

*.Project, .Project
moves only those matching requests entered by any user logged in on the specified project.

., *
moves all matching requests regardless of who entered them.

Access required: The user must have at least "a" extended access to the target queue. If -own (the default) is specified, the user must have at least "o" extended access to the source queues. If -user is specified, the user must have at least "rd" extended access to the source queues and access to the queue_admin_ gate. If the user has AIM ring one privilege, the AIM attributes of the original submitter are preserved; otherwise, the AIM attributes of the current process are used.

Notes on request identifiers: Multiple -id STR request identifiers may be specified on the command line only if no path or -entry STR identifiers are given.

If path or -entry STR request identifiers are given, only one -id STR request identifier may be given in which case only those requests which match one of the path or -entry STR identifiers and which match the -id STR identifier are moved.

If a path or -entry STR request identifier matches more than one request and is not a starname, a message is printed telling how many matching requests were found but none of the requests are moved. The -id STR request identifier may be used to further qualify the path or -entry STR identifier to select the specific request to be moved.

move_imft_request mir

Notes

If the request is already being transferred, this command prints an explanatory message and does not move the request.

If conflicting control arguments (e.g., -long, -brief) are given on the command line, the rightmost control argument takes effect.

Examples

The following examples show three different uses of the mir command, and the responses they generate. Assume that the files s2.compin, s3.compin, s4.compin, and s5.compin are in CISL IMFT queue 2, and s4.compin is already running.

```
mir s2.compin -tq 3
IMFT request s2.compin moved from imft queue 2; ID: 16055.1.
1 request moved; 0 already in imft queue 3.
r:11:04 0.368 1


mir s3.compin -tq 1 -bf
r:11:06 0.467 9


mir -et *.compin -tq 3
move_imft_request: IMFT request s4.compin is already running
  and will not be moved.
IMFT request s5.compin moved from imft queue 2; ID: 160212.9.
1 request moved; 1 already in imft queue 3.
r:11:09 0.347 8
```

Name: print_imft_sites

The print_imft_sites command displays the names of foreign sites that can be used with the -source or -destination control arguments of the enter_imft_request command.

Usage

print_imft_sites

Example

A user is at a site that has IMFT connections to MIT and CISL. He types the print_imft_sites command, and gets the following response:

```
Site name          Access ID          Dest     Source

CISL               IMFT.Daemon.*        X        X
MIT (default)      IMFT.Daemon.*        X        X
```

INDEX

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

CUT ALONG LINE

TITLE: INTER-MULTICS FILE TRANSFER FACILITY REFERENCE MANUAL

ORDER NO. CY73-01

DATED DECEMBER 1983

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

CUT ALONG LINE

FOLD ALONG LINE

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

FOLD ALONG LINE

Honeywell

Together, we can find the answers.

Honeywell

Honeywell Information Systems
U.S.A.: 200 Smith St., MS 486, Waltham, MA 02154
Canada: 155 Gordon Baker Rd., Willowdale, ON M2H 3N7
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH **Italy**: 32 Via Pirelli, 20124 Milano
Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F. **Japan**: 2-2 Kanda Jimbo-cho Chiyoda-ku, Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia**: Mandarin Plaza, Tsimshatsui East, H.K.

37886, 5C983, Printed in U.S.A.

CY73-01