To: MTB Distribution

From: Gary M. Palter

Date: 1 September 1980

Subject: HASP Workstation Simulator

Purpose

This memo describes the Multics HASP workstation simulator facility.

Overview

Several sites have requested the capability to use Multics to submit jobs to and receive output from other computer systems. The majority of these systems can communicate with remote job entry (RJE) stations employing the HASP communications protocol. Thus, an effort was undertaken to permit Multics to simulate a HASP workstation.

One of the major advantages of the HASP protocol over other RJE protocols is that it permits a workstation to have multiple card readers, punches, and line printers all operating simultaneously. To take advantage of this feature on Multics, it is necessary to have a separate process for each device being simulated. However, all communications with the remote system takes place on a single physical channel which normally can be accessed only by a single Multics process. To solve this problem, a ring-0 multiplexer was developed which splits the single channel into separate logical channels for each device allowing for the desired independent operation of simulated devices.

The structure chosen to simulate the devices of the workstation is the I/O daemon structure. A new I/O daemon driver program, similar to the remote_driver_ module, was developed to permit an I/O daemon process to simulate a device of the workstation. Unlike other I/O daemon drivers, however, this driver takes requests from a queue and sends them through a card reader to the remote system; it also receives files from the remote system through card punches and line printers and queues these files for local printing or punching.

A feature was added to the I/O daemon driver to allow output files received from the remote system to be placed into the system pool. When this feature is enabled, users are able to retrieve the output of their foreign jobs and examine it online. In order to use this feature, however, it is necessary to require that special control records, similar to the card input control records in use today, be placed in each output file to identify which Multics users "owns" the file. As these control records must be included by the remote system, requiring possible modifications to the remote system itself,

Multics Project internal working documentation. Not to be reproduced or distributed outside the Multics Project.

the decision was made that the default mode of operation for the simulator would be automatic printing or punching of the received files.

Organization of this Document

The remainder of this document is a draft of the additions to the Bulk I/O manual and the MPM Reference Guide which describe this facility.

APPENDIX G

THE HASP WORKSTATION SIMULATOR

Multics provides a facility for the simulation of a remote job entry (RJE) workstation using the HASP communications protocol. Through this facility, Multics users can request that job decks be transmitted to a remote system for execution and the resulting output be returned to Multics for printing/punching or online perusal.

A HASP workstation is composed of card readers, card punches, line printers, and an operator's console. Each device to be simulated by Multics is configured as a separate sub-channel of a physcial communications channel defined in the CMF as a HASP multiplexer channel. (See MAM Communications for details on configuring a HASP multiplexer.) Up to eight card readers may be configured in a workstation; a total of no more than eight line printers and card punches may be configured; exactly one operator's console must be configured.

Structure of the Simulator

The I/O daemon driver module hasp_ws_sim_driver_ simulates the operation of a workstation's card readers, line printers, and card punches; the command hasp_host_operators_console simulates the console. A separate process is used to simulate each device to permit all devices to operate asynchronously, thus achieving maximum throughput over the communications line.

The simulated operator's console is used to establish the identity of the workstation with the remote system. Subsequently, it may be used to control the operation of the workstation, request status on jobs executing on the remote system, and examine the queues of output files waiting for transmission to Multics.

Card decks are transmitted from Multics through the simulated card readers to the remote system. These decks are normally jobs to be executed by the remote system. On Multics, each card deck must be contained in a segment. A Multics user requests that a deck be transmitted by issuing the dpunch command; a separate request type is used for each remote system.

The remote system transmits output files to Multics through the simulated line printers and card punches. By default, the simulator automatically issues dprint or dpunch requests for these files as appropriate. However, a site may choose to have these output files placed into the system pool storage for subsequent retrieval by Multics users. To use this option, the driver process must be instructed to to expect control records in each output file and the remote system must include these Multics control records to indicate which Multics user owns the file. Adding control records to an output file may involve modifications to the remote computer's operating system, the JCL of each job submitted for remote execution, the programs executed by the each job, or a combination of the above. (See MPM Reference for a description of the format of these control records.)

Defining a HASP Workstation Simulator

To define a workstation simulator, the local administrator(s) must:

o Define the configuration of the workstation being simulated: the number of card readers, line printers, and card punches must be agreed upon with the remote system's administrator(s).

o Determine if the remote system requires that a SIGNON control record be transmitted to establish the identity of the workstation. (The SIGNON record is a special record defined by the HASP protocol to enable the host system to estabslish the identity of the workstation. Many operating systems do not require this control record, but validate the workstation in other ways.) If a SIGNON record is required, it's exact content must be determined for use in the attach descriptions described below.

o Define the HASP multiplexer channel as described in MAM Communications.

o Define a major device for each simulated device except the operator's console and a request type for the submission of card decks in the system iod_tables.

o Create an ACS segment for each sub-channel of the HASP multiplexer channel, give the process which will attach that sub-channel rw access to the ACS and the dialok attribute in the PDT. (See MAM Communications and MAM System.) It is recommended that the process which attaches the simulated operator's console not be registered on the SysDaemon project.

o Determine the printer channel stops used in output files returned from the remote system and insure that the Multics request type(s) used to print those files include the appropriate logical channel stops in their RQTI segments. (See "Request Type Info Segments" in section 2 of this manual.) For example, many systems use channel stop #1 to represent the top of a page; the RQTI segments should specify "Line (1): 1;" to insure correctly formatted output.

IOD_TABLES

With the exception of the operator's console, each simulated device is controlled by an I/O daemon using the hasp_ws_sim_driver_ module. A separate major device with exactly one minor device must be defined in the iod_tables for each simulated device.

The major device definition must include a line statement specifying the sub-channel of the simulated device; the "line: variable;" construct is not allowed. Additionaly, an args statement must be included specifying a station ID and use of the hasp_host_ terminal I/O module (see MPM Communications).

The minor device specification must include a minor_args statement which specifies the type of device being simulated. Additional keywords may be used in this statement as described below.

See "I/O Daemon Tables" in section 2 of this manual for a description of the iod_tables source language.

Sample iod_tables Defintion

The iod_tables entries to simulate a HASP workstation with a card reader, card punch, and two line printers follows:

```
Device:            cdc_rdr1;                    /* Card reader */
 line:             a.h014.rdr1;
 driver_module:    hasp_ws_sim_driver_;
 args:             "station= CDC, desc= -terminal hasp_host_ -comm hasp";
 minor_device:     rdr1;
  minor_args:      "dev= reader_out";
  default_type:    cdc_jobs;

Device:            cdc_prt1;                    /* Line printer #1 */
 line:             a.h014.prt1;
 driver_module:    hasp_ws_sim_driver_;
 args:             "station= CDC, desc= -terminal hasp_host_ -comm hasp";
 minor_device:     prt1;
  minor_args:      "dev= printer_in, request_type= cdc_output";
  default_type:    dummy;

Device:            cdc_prt2;                    /* Line printer #2 */
 line:             a.h014.prt2;
 driver_module:    hasp_ws_sim_driver_;
 args:             "station= CDC, desc= -terminal hasp_host_ -comm hasp";
 minor_device:     prt2;
  minor_args:      "dev= printer_in, auto_queue= no";
  default_type:    dummy;

Device:            cdc_pun1;                    /* Card punch */
 line:             a.h014.pun1;
 driver_module:    hasp_ws_sim_driver_;
 args:             "station= CDC, desc= -terminal hasp_host_ -comm hasp";
 minor_device:     pun1;
  minor_args:      "dev= punch_in";
  default_type:    dummy;
```

```
Request_type:       cdc_jobs;          /* Request type for submitting card */
 generic_type:      punch;             /* ... decks to remote CDC system */
 max_queues:        1;
 device:            cdc_rdr1.rdr1;

Request_type:       dummy;             /* Required by line printers and */
 generic_type:      dummy;             /* ... card punches to avoid errors */
 max_queues:        1;                 /* ... from iod_tables_compiler */
 device:            cdc_prt1.prt1;
 device:            cdc_prt2.prt2;
 device;            cdc_pun1.pun1;
```

args Statement Keywords

station= <station_id>
: identifies returned output files when said files are printed/punched automatically. This keyword is required; the same value should be used for all devices of a workstation simulator.

desc= <attach_description>
: specifies the attach description used to attach the terminal/device I/O module. This keyword is required. The attach description must include the "-terminal hasp_host_" and "-comm hasp" options; the "-tty" option is provided automatically by the driver process. If the remote system requires a SIGNON record, the "-signon" option must be included for all devices of the workstation. (See MPM Communications for a description of the hasp_host_ I/O module.)

minor_args Statement Keywords

dev= <device_type>
: specifies the type of device being simulated by this driver process. This keyword is required. The acceptable values for device_type are:

    reader_out
    : simulates a card reader for sending card decks to the remote system.

    printer_in
    : simulates a line printer for receiving output files from the remote system.

    punch_in
    : simulates a card punch for receiving card decks from the remote system.

auto_queue= <switch_value>
specifies whether output files received by this driver are (1) automatically printed or punched locally or (2) scanned for Multics control records and made available for online perusal as described above. The possible values for switch_value are:

yes
automatically queue the files for printing/punching; do not scan for control records, or

no
scan the output files for Multics control records and store them in system pool storage for online perusal; do not automatically queue files for printing/punching.

This keyword can not be given if "dev= reader_out" is specified. This keyword is optional; the default value is "yes" (automatically queue output files).

request_type= <rqt_name>
rqt= <rqt_name>
specifies the Multics request type to be used for automatically printing or punching output files. The request type specified must be of generic type "printer" if "dev= printer_in" is given or generic type "punch" if "dev= punch_in" is given; this keyword can not be given if "dev= reader_out" is specified. This keyword is optional; the default request type used is the default specified for the appropriate generic type.

Operating a HASP Workstation Simulator

SIMULATOR INITIALIZATION

To start a HASP workstation simulator:

o If necessary, issue the initializer "load_mpx" command described in the MOH to cause the HASP multiplexer channel to wait for a connection.

o Login the process which is to run the simulated operator's console of the workstation and issue the hasp_host_operators_console (hhoc) command, described below, to wait for the connection to be completed. If the remote system requires a SIGNON record as part of the connection procedure, include the "-signon" option on the hhoc command line.

o Complete the physical connection to the remote system.

o When the process running the operator's console prints the message "Input:" indicating that the physical connection is established, perform any logon sequence required to identify the workstation to the remote system. The exact sequence used, if any, should be determined from the remote system's administrative staff.

- o Login each of the driver processes for the other simulated devices. The sequence used to login a driver process is described in "Login and Initialization of Device Drivers" in section 3 of this manual.

- o On the terminal of the process running the operator's console, issue any commands to the remote system required to ready all the devices of the workstation.

- o For each driver process running a simulated card reader, issue the commands:
  ```
  ready
  pun_control autopunch
  go
  ```
  These commands will start the transmission of card decks to the remote system.

- o Issue the "receive" command for each driver process running a simulated line printer or card punch. This command will cause these drivers to wait for output files to be sent by the remote system. As each output file is received, it is processed according to the specifications given in the minor_args statement of the driver as described above.

SPECIAL INSTRUCTIONS FOR RUNNING THE PRINTER AND PUNCH SIMULATORS

In addition to the commands described in this section, the only other I/O daemon commands which may be used in the driver process of a simulated line printer or card punch are: logout, hold, new_device, inactive_time, x, start, help, status, reinit, release, and clean_pool. These commands are described in section 3 of this manual.

After use of the "receive" command described below, the driver only recognizes pending commands while it is between output files. If it is necessary to execute a command while a file is being received, a QUIT must be issued to the driver to bring the driver to QUIT command level. The "hold" command can then be used to cause the driver to remain at QUIT level; the "release" command can be used to abort receiving the file and return to normal command level; and the "start" command can be used to resume receiving the file.

Name: receive

The receive command causes the driver to wait for output files to be transmitted from the remote system. A message is issued at the start and end of each file received. If automatic queueing of output files is enabled for this simulated device, output files will be locally printed or punched after they have been successfully received; otherwise, the output files will be placed into system pool storage as specified by the ++IDENT control records which must be present in the files.

Usage

```
        receive
```

Name: auto_queue

The auto_queue command controls whether output files received by this driver are (1) automatically printed or punched locally or (2) scanned for Multics control records and placed in system pool storage for online perusal.

Usage

```
        auto_queue <switch_value>
```

where:

1. switch_value
   must be chosen from:

   yes
   automatically queue the files for printing/punching; do not scan for control records, or

   no
   scan the output files for Multics control records and store them in system pool storage for online perusal; do not

automatically queue files for printing/punching.

request_type

Name: request_type, rqt

The request_type command is used to specify the request type to be used for the automatic queuing of output files received by this device.

Usage

```
rqt <rqt_name>
```

where:

1. rqt_name
   is the name of the request type to be used for automatic queuing. The generic type of this request type must agree with the type of device being simulated ("printer" for simulated line printers, etc). This parameter is optional; the default value is the request type specified in the iod_tables definition of this driver.

SIMULATOR SHUTDOWN

To shutdown a HASP workstation simulator:

- o Issue the "halt" command for each process running a simulated card reader. This command will cause these drivers to stop transmitting card decks after the current deck (if any) has completed transmission.
- o On the terminal of the process simulating the operator's console, issue any commands to the remote system to request it to stop transmitting additional output files.

- When all driver processes are idle, issue the "logout" command to each driver.

- On the terminal of the process simulating the operator's console, issue any commands to the remote system to indicate that this workstation is signing off. After giving these commands, issue the "!quit" request to the hhoc program and logout the console process.

- Break the physical connection.

- If desired, shutdown the HASP multiplexer by using the initializer "dump_mpx" command described in the MOH.

Name: hasp_host_operators_console, hhoc

The hasp_host_operators_console command is used to simulate the operation of the operator's console of a HASP workstation. The operator's console is used to identify a workstation to a remote system, to issue commands governing the operation of the workstation, and to receive status information from the remote system.

Usage

```
          hhoc tty_channel {attach_arguments}
```

where:

1. tty_channel
   is the name of the terminal channel to be attached as the operator's console. This channel must be configured as the console sub-channel of a HASP multiplexer channel (eg: a.h014.op). See MAM Communications for a further description of the HASP multiplexer.

2. attach_arguments
   are options acceptable to the hasp_host_ I/O module. This command supplies the -comm, -tty, and -device options automatically; these options need not be given on the command line. (See MPM Communications for a description of the hasp_host_ I/O module.)

Notes

If the remote system requires a SIGNON record be transmitted before normal operations of a workstation may commence, the -signon option should be supplied on the command line specifying the exact SIGNON record to be transmitted.

For example, the command line:

```
     hhoc a.h014.opr -signon "/*SIGNON REMOTE7"
```

may be used to attach the channel a.h014.opr as the operator's console of a remote IBM system expecting a connection from the workstation named REMOTE7.

After attaching the channel specified on the command line, hasp_host_operators_console prompts the user for terminal input with the string "Input:".

Input from the terminal is transmitted directly to the remote system unless the line begins with the request character, an exclamation mark (!); lines beginning with the request character are interpreted by this command. The valid requests are described below.

Any text received from the remote system is displayed directly on the terminal without any interpretation by hasp_host_operators_console.

HASP_HOST_OPERATORS_CONSOLE REQUESTS

The following requests are recognized by hasp_host_operators_console when given at the beginning of a line of terminal input:

!.. <REST_OF_LINE>
: the rest of the line is passed to the Multics command processor for execution as ordinary commands.

!.
: prints a message of the form:

```
hasp_host_operators_console N.N; connected to channel NAME.
```

where N.N is the current version of this program and NAME identifies the channel connected as a console to the remote system.

!quit
: causes the command to hangup the operator's console channel and return to Multics command level.

SECTION 5

INPUT AND OUTPUT FACILITIES

RJE WITH FOREIGN COMPUTER SYSTEMS

Multics provides facilities for users to submit card decks to a remote computer system for execution and to receive output from that execution for either printing/punching locally or online perusal. This section describes the mechanisms available for using this facility.

Submitting Card Decks to a Remote System

Each card deck to be transmitted to a remote system for execution must be contained in a separate Multics segment. This segment can be created using an editor, bulk card input, or any other appropriate mechanism.

The segment must consist of ASCII text only; no binary data (object segments, etc.) may be included. The exact format of the contents of the segment is dependent on the remote system being accessed and should be determined from the appropriate documentation for the remote system.

To transmit the segment to the remote system, issue the dpunch command specifying the mcc conversion mode and the request type established by your system administrator(s) explicitly for this purpose. A separate request type will be used for each remote system to which card decks can be submitted.

For example, to submit the card deck contained in the segment "sample.cdc" in the working directory to a remote CDC system, deleting the deck after it is successfully transmitted, issue the command:

```
    dpunch -mcc -rqt cdc_jobs -dl sample.cdc
```

where "cdc_jobs" is the request type established by your system administrator(s) to submit decks to the CDC system.

Receiving Output from a Remote System

By default, printed and punched output returned by a remote system to Multics is automatically printed or punched locally. However, your system administrator(s) may decide that the returned output should be made available to users for online perusal.

If output is to be available for online perusal, each output file must contain Multics control records which establish the identity of the user who owns the file. Either the job control language (JCL) submitted to the remote system or the program(s) executed on the remote system must be modified to cause the required control records to appear in the output files. Check with your system administrator(s) to determine which mechanism must be used for each remote system.

Returned output files which are to be available for online perusal are placed in system pool storage where they may be retrieved using the copy_cards command described in MPM Commands. Output files must be copied in a reasonable time, as they are periodically deleted from the system pool.

Format of an Output File Transmitted to Multics for Online Perusal

```
++IDENT FILE_NAME PERSON_ID PROJECT_ID
++FORMAT MODES
++CONTROL OVERWRITE AUTO_QUEUE
++INPUT
   .
   .
(output data)
   .
   .
<EOF record>
```

The only user-supplied control records required are ++IDENT and ++INPUT. For an explanation of these control records, refer to Appendix H of this manual.

Each output file is delimited by an end-of-file (EOF) record supplied automatically by the remote system. All control records in the output file from ++IDENT through ++INPUT inclusive and the EOF record are removed from the file before it is placed into pool storage.

For printed output, each paper motion command in the file is translated into the character sequence which will best simulate the requested motion when (and if) the file is printed locally via the dprint command. The exact character sequences used are given in Table 5-1.

One of the paper motion commands that may be received is a request to skip to a specific printer channel stop. This command is converted to a logical channel slew sequence as defined in "Vertical Format Control" earlier in this section. The user should check the RQTI segment of the request type used for printing the output file to determine which channel stops may be used in the output file. (The program executed on the remote system is responsible for placing this particular paper motion command in the output file. The exact mechanism used to do this should be determined from the appropriate documentation for the remote system.)

Table 5-1: Translations of Paper Motion Commands in Output Files

Paper motion command	Character sequence
Slew zero lines	CR (octal 015) (1)
Slew one line	NL (octal 012)
Skip to channel N	ESC c <N> ETX (2)

(1) Overprint the current line with the previous line.

(2) This sequence is octal 033, octal 143, the decimal representation of the channel number encoded as ASCII characters (eg: octal 061, octal 065 for channel #15), and octal 003.

APPENDIX H

RETURNED OUTPUT CONTROL RECORDS

This appendix defines the control records which are permitted in output files returned by a remote system to Multics for online perusal by Multics users.

All characters on a control record are converted to lower case except those immediately following the escape character (backslash, \). For example, \SMITH.\SYS\MAINT is mapped into Smith.SysMaint.

Control record format is as follows:

- o Columns one and two contain ++,
- o A keyword appears starting in column three,
- o Remainder of the record is free form,
- o Continuation of control records is not permitted; the entire record must be contained within one punch or printer record, and

++IDENT

<u>Name</u>: ++IDENT

This control record identifies the Multics user who is to receive the output file. All records in the output file before the ++IDENT control record are discarded. All three fields of this control record must be specified in the order shown.

<u>Usage</u>

```
++IDENT <FILE_NAME> <PERSON_ID> <PROJECT_ID>
```

where:

1. FILE_NAME
   is the name used to identify the output file in system pool storage. It should be unique among the user's output files recently received. In the event of name duplications, the system output receiving

process appends a numeric component to the end of the supplied name and creates a duplicate segment for FILE_NAME unless the OVERWRITE control option is specified on the ++CONTROL record.

2. PERSON_ID
   is the registered person name of the owner of this output file. Only this person is able to copy the file from the pool.

3. PROJECT_ID
   is the registered project name of the owner.

Notes

Multics person and project names normally begin with uppercase letters. Such names must have the escape character before each uppercase letter, since all letters in a control record are mapped to lowercase except those immediately following the escape character (backslash).

Angle brackets in the "Usage" line indicate information supplied by the user.

++CONTROL

Name: ++CONTROL

This control record is used to modify the operation of the output file receiving software. This record is optional.

Usage

```
++CONTROL <CTL_KEYS>
```

where:
1. CTL_KEYS
   specifies the operating modes of the software and may be one of the following:

   OVERWRITE
      specifies that if an output file already exists with the name

given on the ++IDENT control record, the old file is to be deleted before the new file is received. The default action is described under the FILE_NAME argument of the ++IDENT control record.

AUTO_QUEUE
specifies that the output file is to be automatically queued for printing or punching locally as appropriate. The default action is to not queue the file.

REQUEST_TYPE <RQT_NAME>
RQT <RQT_NAME>
specifies use of the RQT_NAME print/punch queue if this output file is automatically printed/punched. RQT_NAME must identify a request type whose generic type is "printer" for print files and "punch" for punch files. (See the description of print_request_types in MPM Commands.) If this ctl_key is not given and automatic queuing is requested, the request type established by the system administrator(s) for output from this remote system will be used. This ctl_key is ignored unless the AUTO_QUEUE ctl_key is also given.

Name: ++FORMAT

This control record is used to specify the conversion modes used to format the data in the output file. This record is optional.

Usage

```
        ++FORMAT <MODES>
```

where:

1. MODES
may be any of the following modes. The meaning of these modes is discussed in "Card Conversion Modes" in Appendix C.

TRIM	
NOTRIM	(default)
LOWERCASE	
NOCONVERT	(default)
ADDNL	
NOADDNL	(default)
CONTIN	
NOCONTIN	(default)

++INPUT

Name: ++INPUT

This control record marks the end of the control records and is required for all output files. The next record is the first record of the user's output file to be placed into system pool storage.

Usage

```
          ++INPUT
```

There are no fields following the key on this control record.

Notes

The system treats all records received after the ++INPUT record as data and places them into the output file even if they have control record syntax as described above.