Published: 05/06/68

Identification

Line Configuration Commands
set_line, get_line
W. R. Strickler, K. J. Martin

Purpose

The system operator must be able to specify and to examine the status of communication lines, which are controlled by the Answering Service process (see BQ.2.01, BQ.2.02) in the system control process-group (see BQ.1.01). The command get_line is used by the operator to examine status of lines, and the command set_line is used to specify line status.

Specify Line Configuration

The command, set_line, is used to specify over which lines users are allowed to dial up and log in to the system and which lines are unavailable. A line is in one of five states:

1. in-use - another user may not dial up
2. on-hook - user may dial up
3. off-hook - user may not dial up
4. no-answer - user may not dial up
5. disabled - out of service

The operator does not specify that a line should be placed in the in-use state. Rather, a line is in in-use state because some user has previously dialed up on it and is still using it. The operator may explicitly specify that a line or a group of lines be placed in any of the other four states.

Line configuration may be specified in three ways:

1. by the Registry File Name (see BT.1) of each communication line and the state it is to assume;
2. by line type, the number of lines of that type to be affected, and the state they are to assume;
3. by stating that all or none of the communication lines are to assume a certain state.

Usage

set_line (line_group1 n1 state1 -line_group2 n2 state2- ...)

where

line_groupi — is either a Registry File Name, a line type, "all" or "none". A Registry File Name is identified by "rfn_" as the first four characters.

ni — is the number of the specified line type to be affected; following a Registry File Name, "all", or "none", ni must be the null character string.

statei — is the state which the specified line(s) will assume. state may be "on", "off", "na", or "dis" for on-hook, off-hook, no-answer, or disabled.

Exceptions (that is, all of set A except ...) may be expressed by specifying first the larger set and its state, then each exception and its state. The requests are processed in the order in which they appear (see implementation) so that the larger set is first processed and then changed for the exceptions.

Implementation

Briefly, set_line places the information contained in the array of arguments into a data segment common to both System Control and the operator's working process, then sends an event to System Control and waits for a reflection signal from System Control indicating that the request has been processed (by answering Service process in the System Control process-group).

Information telling System Control what is to be done is placed in the segment, "set_line", in the request directory of System Control. The information is stored in a structure with the following declaration:

```
dcl  1 line_set based (lp),
     2 request bit (1),              /*"1"b for set_line,
                                       "0"b for get_line*/
     2 n_args fixed bin (17),
     2 args (p->line_set.n_args),
       3 switch bit (1),             /*"1"b if Registry
                                       File Name,
                                       "0"b otherwise*/
       3 line_name char (32),        /*as in arg list*/
       3 n fixed bin (17),           /*as in arg list*/
       3 state bit (3);              /*"101"b - disabled
                                       "100"b - no-answer
                                       "011"b - off-hook
                                       "010"b - on-hook
                                       "001"b - in-use */
```

This structure is used for sending information to and receiving information from System Control by both the set_line and get_line commands.

The command procedure set_line does the following:

1. Creates an event channel over which it can receive an event from System Control signaling completion of the request; the name of the event channel is placed in rp->op_req.ref_chn, in the segment "request_name" (see BX.15.05).

2. Places the name "set_line" in rp->op_req.req_name.

3. Stores the information from the argument list into the appropriate elements of the structure line_set and sets lp->line_set.request to "1"b.

   (Never sets lp->line_set.args.state to "001"b, since in-use is not a valid state for the operator to specify).

4. Signals System Control (by calling hcs_$set_event) over the channel named in the structure element p->op_comm.op_req_chn in segment "operator_comm" (see BX.15.05).

5. Calls the Wait Coordinator (BQ.6.05) and waits for a signal over the channel created in step 1. During wait, System Control brings the configuration module of the Answering Service into action. It attempts to process each request. As a request is successfully processed, the module sets the corresponding element lp→line_set.args(i).state to "000"b.

6. When set_line receives a wakeup, it examines each element lp→line_set.args(i).state to see if it is non-zero. The set_line command prints out for the operator either a statement that configuration is complete as requested or the details of any unsatisfied requests (those for which state is not zero).

Request Line Configuration Status

The command, get_line, is used to find out the configuration status of any combination of communication lines. Status may be requested in three ways:

1. Specify the Registry File Names of each communication line for which status is desired;

2. Specify the line type for which the operator wants status of all lines;

3. Specify that status information for all lines is wanted.

Usage

```
get_line (line_group1 -line_group2- ...)
```

where

line_groupi is either a Registry File Name, a line type, or "all".

Status is printed out by line group in the order requested. When a line type or "all" is requested, status of individual Registry File Names is given if members of a line type are in several different states. If all members of a line type are in the same state, Registry File Names are not listed.

Implementation

The command get_line uses the same data segments as set_line, and follows essentially the same setup:

1. Creates an event channel over which System Control can signal.

2. Places "get_line" in rp->op_req.req_name.

3. Stores information from the array arguments into the structure line_set, and sets lp->line_set.request to "0"b.

4. Signals System Control.

5. Calls the Wait Coordinator.

6. On wakeup, after the Answering Service process in System Control has placed the results (status of communication lines or groups of lines) into the line_set. args substructure, get_line takes this information from the substructure, formats it, and prints it at the operator's console.