Identification

Operator Commands to the Answering Service

configure, line_status

K.J. Martin

Purpose

The system operator must be able to specify and to examine the status of communication lines. Since communication lines are controlled by the Answering Service process (see BQ.2.01, .02) commands are provided which inform the Answering Service of the operator's wishes and request information from the Answering Service. The commands, configure and line_status, are described below.

Specify Line Configuration

The command, configure, 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 the 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, but no one will answer.
5. disabled - out of service

The operator does not specify that a line should be placed in 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. specify the Registry File Name (see BT.1) of each communication line and the state it is to assume;
2. specify a line type, the number of lines of that type to be affected, and the state they are to assume;
3. specify that all or none of the communication lines are to assume a certain state.

Usage

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

where

line_groupi is either a Registry File Name, a line type, "all", or "none".

-ni- is the number of the specified line type to be affected. n is not appropriate following a Registry File Name, "all", or "none".

statei is the state which the specified line(s) are to 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 then the exceptions changed.

Implementation

Briefly, configure places the information contained in the array of arguments into a data segment common to both the Answering Service process and the operator's Working process, then sends an event to the Answering Service and waits for a specified event from the Answering Service indicating that the request has been processed. The event to the Answering Service is received on an event call channel (see BQ.6.03) and causes the configuration module (see BQ.2.02) to be called in the Answering Service. This module reads the data segment and effects the changes called for, then sends the awaited event to the operator's Working process. If any requests could not be filled, the module places that information in the same data segment. The remaining details are: specifications for the data segment, and how configure finds the information necessary to send an event to the Answering Service.

After the operator logs in, the Answering Service places its own process id and the event channel id of the event call to its configuration module in the segment, a_s_communication, which is in the process directory of the operator's Overseer process. Configure can access them directly to make the call to set_event (BQ.6.02).

Information telling the Answering Service what is to be done is placed in the segment, line_request, which is common to both processes. The information is stored in a structure with the following declaration:

```
dcl 1 operator_commands ctl (p),
    2 process_id bit (36),          /* of operator's working process */
    2 event_id bit (70),            /* to signal completion of requested action */
    2 request bit (1),              /* configure if "1"b, get the line status
                                       if "0"b  */
    2 n_args fixed bin (17),
    2 arguments (p → operator_commands.n_args),
```

```
3 line_group,              /* as in arg list */
  4 n_chars fixed bin (17),
  4 name char (p → operator_commands.arguments.line_group.n_chars),
3 n fixed bin (17),        /* as in arg list */
3 state bit (3);           /* "100"b - disabled,
                              "000"b - no-answer
                              "001"b - off-hook,
                              "010"b - on-hook,
                              "011"b - in-use   */
```

This structure is used for sending information to and receiving information from the Answering Service for both the configure and line_status commands.

Configure creates an event channel over which it can receive an event from the Answering Service, then stores the information from the argument list into the appropriate structure elements and sets p → operator_commands.request to "1"b. The element p → operator_commands.argument.state is never set to "011"b by configure since in-use is not a valid state for the operator to specify.

The configuration module attempts to process each request. As a request is successfully processed, the module sets all elements of the substructure p → operator_commands.arguments to null or zero. Thus when the configuration module sends an event to the operator's working process, only those requests which could not be satisfied remain in the structure. The configure command prints out for the operator the details of any unsatisfied requests.

Request Line Configuration Status

The command, line_status, 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 Name of each communications 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

line_status (line_group1 line_group 2...)

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, individual Registry File Names are not listed.

Implementation

Line_status places the information from the array of arguments into the structure operator_commands, in the segment, line_request. It creates an event channel over which it can receive an event from the Answering Service, then stores that event id and its own process id in the structure. The element p → operator_commands. request is set to "0"b to indicate that the command at hand is line_status. Line_status then sends an event to the Answering Service and waits for an event from the Answering Service.

The Answering Service processes the arguments, placing the results (i.e., the status of the communications lines or groups of lines) back into the structure, operator_commands, in the segment, line_request. When all results are in the structure, the Answering Service sends an event back to the operator's process as specified by the process id and event id in the operator_commands structure.

Line_status takes the information from the structure, formats it and prints it at the operator's console.