NCR CONTROL DATA CORPORATION

ADVANCED SYSTEMS LABORATORY

CHAPTER 12

OPERATOR COMMUNICATIONS

Doc. No. ASL00282

Rev. 04

Copy No. 87

NCR/CDC PRIVATE

TABLE OF CONTENTS - OPERATOR COMMUNICATIONS

1.0 INTRODUCTION

This document is the GDS for the Operator Communications System (later referred to as OCS) for IPL/OS.

The functions described are the basic capabilities of the system. As the OS requirements for operator communications services become better defined, these capabilities will be extended. It is expected that most of the additional capability will be in the form of additional commands.

2.0 GENERAL SYSTEM DESCRIPTION

2.1 DESIGN OBJECTIVES

The primary objective of the Operator Communications System (OCS) is to provide a facility whereby both the system and users may converse with the human operator(s) of the system or network. Specific objectives of this design are as follows.

- The system should be flexible and easily extended at the site.
- The system will have several logical operators, each with different responsibilities and privileges.
- The set of logical operators may be mapped onto one or more physical operators.
- The hardware and software necessary to support the input and output streams of a physical operator should be no different from that of any other job. If special hardware is required for specific functions, such as a graphics terminal for performance displays, that requirement would be satisfied in the same way as a user job that must use such a terminal.
- The system should be capable of operating in a default mode without any physical operators.
- A site in a network may be operated by a physical operator at a different site. This is not an initial release objective.

2.2 CLASSES OF OPERATORS

Seven logical operators with five levels of privilege are defined for each site. With the exception of REMOTE#OP, which is an alias for the user logged in on the remote batch terminal, each logical operator may be logged in from only one origin at a time. Their responsibilities and privileges are as follows.

2.2.1 CE#OP

This is the customer engineer. He or she is responsible for the proper function of the system hardware. CE#OP has the highest level of privilege of the operators and is capable of command functions such as taking full control of a processor that is recognized as "up" by the operating system.

2.2.2 SE#OP

This is the systems engineer. He or she is responsible for the proper function of the system software. SE#OP has the second highest level of privilege and is capable of command functions such as replacing system modules and modifying the contents of system tables while the system is running.

2.2.3 SYSTEM#OP

This is the main console operator. He or she is responsible for "driving" the system and monitoring its normal operation. SYSTEM#OP has the third highest level of privilege and is capable of command functions such as starting, stopping or cancelling jobs and generally modifying the systems operational parameters.

2.2.4 MASS#OP, TAPE#OP, BATCH#OP

These are the operators responsible for servicing user or operating system requests for attention to mass storage, magnetic tape, batch and telecommunications devices. They have the fourth highest level of privilege and are capable of command functions such as rendering a device offline or online and directing output spooling.

2.2.5 REMOTE#OP

This is the operator of the remote batch terminal associated with a job. He or she is responsible for servicing user or operating system requests for attention to components of the remote batch configuration. REMOTE#OP has the lowest level of privilege of any operator and is only permitted command functions affecting the local status of the remote batch terminal.

2.3 IMPLEMENTATION STRATEGY

The Operator Communications System (OCS) is composed of a set of jobs each representing a physical operator of the system. The primary task of each of these jobs is the Sequence Monitor task which includes the System Command Language interpreter (SCL). In addition to the standard SCL, certain signals are selected to a queue. Interrupt procedures are added to process these signals. The primary purpose of this signal selection is to receive operator messages generated by OCS requests issued in other jobs. The text of a message is contained within the signal information.

Each operators environment will be built by the SCL user profile. In general, the profile will validate the operator, declare an OCS queue, select events and allow interrupt procedures, attach the LNS segment containing the operators repertoire and update the alternate operator routing information.

All other jobs must have an OCS queue and select the OCS signal id during job initiation. These functions will be performed by the OCS#INIT procedure.

The decision to represent each physical operator as an independent job rather than as a task of the system job is based on the following advantages.

- The standard sequence monitor and SCL interpreter code may be used to interface to the input and output streams. Therefore no special logic is required to handle the I/O interface to OCS and all modes of physical access to the system are automatically supported. Also, OCS code will be insensitive to changes in SCL.
- The normal user and system repertoires will be available to operators, thus extending the operators' capability and eliminating the need for special operator commands which approximate normal user functions.
- Each physical operator will have a unique LNS search list and local LNS segment.
- Signals do not require redistribution within the

system job to operator tasks.

- REMOTE#OP is a special representation of the remote batch user and is already a job by definition.

- By utilizing the OS facilities available to jobs, less code is required to satisfy OCS requirements.

It is planned to begin implementation with the Job Interface (task services code) and the Job Communication commands when the required Program Management facilities are available. The individual commands of the repertoires will be implemented gradually as the methods of achieving each function within the capabilities of OS are determined.

2.4 DAYFILE RECORDING

All messages to an OCS operator job will be written to the IOC#OUTPUT stream of the particular OCS operator job. Messages from an OCS operator job will be written to either the IOC#INPUT or IOC#ALTERNATE streams depending on the origin of the command. This function is the standard operation of the SCL interpreter.

Messages to operators from other jobs and operator input to these jobs will be written to the OCS#LOG stream of the job by OCS task services.

3.0 JOB INTERFACE

The job interface is accomplished via OCS control language requests. Each time operator input is received by an OCS operator job, a signal is sent to the appropriate job with the signal id of OCS. This pre-selected signal is placed in the jobs OCS queue for processing by the OC#RECEIVE request.

When a job wishes to direct a message to an operator, the OCS task service request sends a signal to the appropriate OCS operator job. An interrupt procedure will be activated in the OCS operator job to process the message.

If the operator with which a job wishes to converse is not logged in to the system, the information will be passed to the alternate for that particular operator. Conversely, if messages are expected from a particular logical operator, they may be received from an alternate. The user or system component must consider this when using OCS for a function that is sensitive to the actions of a particular operator.

3.1 OC#CONVERSE

The purpose of the OC#CONVERSE request is to send a message to an operator and solicit a reply. The request does not wait for the reply before returning control.

OC#CONVERSE (text, operator, reply_ordinal, status)

text: The text parameter specifies a string containing the text of the message.

operator: The operator parameter specifies a string containing the name of the logical operator to which the message is to be directed. If this parameter is omitted (indicated by a blank string) the message will be sent to SYSTEM#OP for a local job or to REMOTE#OP for a remote batch job.

reply_ordinal: The reply_ordinal parameter specifies an integer variable into which the reply ordinal generated by OCS is to be placed.

status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".

3.2 OC#INFORM

The purpose of the OC#INFORM request is to send an information message to an operator. No reply is expected.

OC#INFORM (text, operator, status)

text: The text parameter specifies a string containing the text of the message.

operator: The operator parameter specifies a string containing the name of the logical operator to which the message is to be directed. If this parameter is omitted (indicated by a blank string) the message will be sent to SYSTEM#OP for a local job or to REMOTE#OP for a remote batch job.

status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".

3.3 OC#RECEIVE

The purpose of the OC#RECEIVE request is to accept the text of operator input.

OC#RECEIVE (wait, operator, reply_ordinal, buffer, count, status)

wait: The wait parameter specifies a boolean variable. If the value of the variable is true, control will not be returned until a message meeting the criteria of the other parameters is received. If the value of the variable is false, control will be returned immediately with a queued message or an appropriate status.

operator: The operator parameter specifies a string containing the name of the logical operator from which the message must be received. If this parameter is omitted (indicated by a blank string), input from any operator will satisfy the request.

reply_ordinal: The reply_ordinal parameter specifies an integer containing the reply ordinal returned by the OC#CONVERSE request for which a reply must be received. If this parameter is omitted (indicated by a zero), any input will satisfy the request.

buffer: The buffer parameter specifies a string variable into which the text of the message is to be placed.

count: The count parameter specifies an integer variable into which the number of outstanding messages not yet received by the job is to be placed.

status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".

NOTE: The conditional relationship between the operator and the reply_ordinal parameters is "logical and". That

is, if both are specified, only a message meeting both criteria will be received.

3.4 OC#SELECT

The purpose of the OC#SELECT request is to associate an event control block with the arrival of operator input. This event control block is user defined and is secondary to the one internally associated with the task service OCS queue of signals. After the OC#SELECT request is issued the OCS interrupt procedure within the job will issue a PM#CAUSE_EVENT on the specified event control block.

This request is intended for optional use to reduce the number of unproductive executions of OC#RECEIVE in a program that must allow for unsolicited operator input. Refer to the Program Management GDS for a description of the event control block and its uses.

OC#SELECT (event, status)

event: The event parameter specifies a record containing an event control block to be associated with the arrival of operator messages.

status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".

4.0 SYSTEM INTERFACE

The system interface to OCS may be of four types. First, the requests described in "3.0 Job Interface" may be issued by a system job or by task services on behalf of a job. Second, an operator command will execute an SCL command procedure that may issue one or more task service requests on behalf of the OCS job itself. Third, a command procedure may invoke a new task or subtask within the OCS job. Fourth, a command procedure may send one or more signals to other jobs.

4.1 REQUESTS TO TASK SERVICES

In addition to the normal user and system repertoires, each logical operator has its own commands. Most of these commands result in a request being issued to task services for the required function to be performed. For example, the operator command to bring a device online would result in task service requests to LNS and the I/O system to construct the required tables and open the device.

4.2 ASYNCHRONOUS FUNCTIONS

Some command functions may be best achieved by the execution of an asynchronous task or subtask. In this case the command procedure will issue a PM#EXECUTE request to invoke the required program. Examples of this type of command would be to begin collecting measurement data or to run an on-line diagnostic.

4.3 SIGNALS TO JOBS

Certain operator commands may require the interruption of jobs in the system. This is accomplished by the command procedure issuing a request to send a signal to the job. Although the command procedure is similar to a direct request in that a PM#SEND_SIGNAL request is issued, the receiving job must have selected the signal.

Two examples of this type of command are the commands to cancel a job, (where the sequence monitor of the job has selected the cancel signal) and the command to stop or restart a listing currently being printed. In the second case, several signals of this type are selected by File Router.

5.0 OPERATOR INTERFACE

The operator interface is achieved by interacting with an instance of execution of the System Command Language interpreter (SCL) with the added capability of receiving unsolicited output to satisfy the requirements of system-to-operator communication. An operator is made known to the system via the normal LOGIN functions and is established as an independent job.

The operator interface is expected to evolve in response to both system and operational requirements. As needs arise for various system components to interact with operators, the appropriate commands will be added to the repertoire. Also, operating requirements may require extensions to the operating system in the form of requests or sensitivity to signals.

In addition to the commands described in this section, the full capabilities of SCL including macros, enter files and control constructs are available to all operators. Refer to the System Command Language GDS for further information.

5.1 ALTERNATE OPERATORS

In the event that a job requests communication with a logical operator that is not currently logged in to the system, a system of "buck passing" is required. The alternate for a logical operator must always have the level of privilege required to respond to or to act on the request. This hierarchy of alternates is dynamic and is recorded in global LNS by the system profile at autoload. A recommended system of alternates is described below.

OPERATOR	ALTERNATE
REMOTE#OP	BATCH#OP
BATCH#OP	TAPE#OP
TAPE#OP	MASS#OP
MASS#OP	SYSTEM#OP
SYSTEM#OP	SE#OP
SE#OP	Error to SYSTEM#OP
CE#OP	Error to SYSTEM#OP

If a request was redirected to an alternate operator or no alternate could be found, an appropriate status is returned by the OC#CONVERSE and the OC#INFORM requests.

This rerouting may be accomplished by OCS task service request processors consulting a list of logged in operators in global or shared LNS.

5.2 MESSAGE FORMATS

Messages to operators have the following general format.

[<reply ordinal>].<job id>/<text>.

For example, the output from an OC#CONVERSE request would appear as follows.

3.TAPEJOB/MOUNT TAPE 123456.

The output from an OC#INFORM request would omit the reply ordinal and appear as follows.

LONGJOB/THIS JOB WILL RUN FOR 6 HOURS.

5.3 COMMAND REPERTOIRES

The command repertoires of each level of privilege are contained in LNS segments that are attached when an operator is logged in. Each operator has available the repertoire of its own level of privilege as well as the repertoires of all lower levels. For example, SYSTEM#OP has available SYSTEM#OP, MASS#OP, TAPE#OP, BATCH#OP, REMOTE#OP and job communication repertoires. However, SYSTEM#OP may not use the commands available only to SE#OP and CE#OP.

5.3.1 JOB COMMUNICATION COMMANDS

Job communication commands are used to interact with the OCS task service requests OC#CONVERSE, OC#INFORM and OC#RECEIVE. These commands are similar to those described under "4.3 Signals to System Jobs" except that a standard signal with input text is sent to a possibly non-system job and processed in that job by OCS task services. The job communication commands are available to all operators.

5.3.1.1 Reply

The purpose of the reply command is to respond to a message issued by an OC#CONVERSE request. The syntax is as follows.

REPLY ! R to= <expr> [text= <expr>] [status= <ref name>]

to: The to parameter specifies the reply ordinal associated with the message receiving the response.

text: The text parameter specifies a string containing the text of the reply. Omission of the text parameter will cause the reply to be a blank string.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Examples:

```
reply to=3,text="go"
r 3
```


5.3.1.2 Inform or Interrupt

The purpose of the inform command is to send unsolicited input to a job. The syntax is as follows.

INFORM ! INTERRUPT ! I [to= <ref name>] [text= <expr>] [status= <ref name>]

to: The to parameter specifies the name of the job to which the message is to be sent. Omission of this parameter will cause the message to be broadcast to all jobs.

text: The text parameter specifies a string containing the text of the message to be sent. Omission of the text parameter will cause the text to be a blank string.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Examples:

```
interrupt msgswitch
8.MSGSWITCH/ENTER INSTRUCTIONS.

i text="system going down in 5 min."
```

5.3.1.3 Display

The display command may be used by the operator to display outstanding messages awaiting replies. Refer to the user repertoire for further details. The syntax in this context is as follows.

DISPLAY name= <ref name> [status= <ref name>]

name ! n: This parameter specifies the name of the list of messages to be displayed.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
display batch#op->messages
13.13.13 7.ROUTER/MOUNT FORM ABC ON SYSPRT4.
13.20.01 11.ROUTER/END OF FORMS ON SYSPRT2.
```

Note: the leftmost set of three numbers is the time of day (hh.mm.ss) that the message was issued. The next number is the reply ordinal.

5.3.2 CE#OP REPERTOIRE

5.3.2.1 Status

The purpose of the status command is to display the status of an entity in the system. The command syntax is as follows.

STATUS ! S name= <ref name> [status= <ref name>]

name ! n: This parameter specifies the name of the entity for which a formatted status report is to be displayed. The LNS type of the name indicates the procedure to be used for data collection and formatting. If there is no predetermined format for a particular type the command degenerates to a DISPLAY operation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
s tape100
VOLUME 123456 ON TAPE100.
ASSIGNED TO TAPEJOB.
17 RECOVERED READ ERRORS.
```

5.3.2.2 Seize

The purpose of the seize command is to take over complete control of a hardware resource. In certain cases this may take a considerable amount of time. Therefore, the command may be either synchronous or asynchronous depending in the resource being seized. The command syntax is as follows.

SEIZE ! SZ resource= <ref name> [status= <ref name>]

resource ! r: This parameter specifies the resource to be seized.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Examples:

```
seize resource=cpu1
CONFIGMGR/CPU1 AVAILABLE.

sz disk200
SEIZE ON DISK200 PENDING.
.
 .
  .
CONFIGMGR/DISK200 AVAILABLE.
```


5.3.2.3 Unseize

The purpose of the unseize command is to return the use of a seized resource to the system. The command syntax is as follows.

UNSEIZE ! UZ resource= <ref name> [status= <ref name>]

resource ! r: This parameter specifies the resource to be returned to the system.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
uz core2
```

5.3.3 SE#OP REPERTOIRE

5.3.3.1 Status

The purpose of the status command is to display the status of an entity in the system. The command syntax is as follows.

```
STATUS ! S name= <ref name> [status= <ref name>]
```

name ! n: This parameter specifies the name of the entity for which a formatted status report is to be displayed. The LNS type of the name indicates the procedure to be used for data collection and formatting. If there is no predetermined format for a particular type the command degenerates to a "list data" operation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
s tape100
VOLUME 123456 ON TAPE100.
ASSIGNED TO TAPEJOB.
17 RECOVERED READ ERRORS.
```


5.3.3.2 Memory

The purpose of the memory command is to display or alter either real or virtual memory. The command syntax is as follows.

```
MEMORY ! M real!virtual= <expr> [words= <expr>] [data=
         (<expr>[,<expr>]...)] [status= <ref name>]
```

real ! r ! virtual ! v: This parameter specifies the address to begin memory access. The keyword used indicates whether the address is real or virtual. Omission of the keyword causes a default of virtual to be assumed. The virtual address refers to the address space of the SE#OP operator job and may include an image of System Monitor segments.

words ! w: This parameter specifes the number of words to access. Omission of this parameter will cause a default of one (1) word to be assumed.

data ! d: This parameter specifies the data to be entered into the memory described. Omission of this parameter will cause the memory to be displayed in hexadecimal and ascii representation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Examples:

```
memory loc(tape100)
SSSS AAAAAAAAAAAA  5441504531303020  TAPE100
```

(Where SSSS is the segment and AAAAAAAAAAAA is the address.)

```
m r=1000(16),2,(5,0e(16))
0000 000000001000  0000000000000005  ........
0000 000000001008  000000000000000E  ........
```

5.3.3.3 Patch

The purpose of the patch command is to dynamically replace one or more procedures in the system. The command syntax is as follows.

PATCH ! P proc= (<ident>[,<ident>]...) [job= (<ref name>[,<ref name>]...)] [status= <ref name>]

procedure ! procedures ! proc ! procs ! p: This parameter specifies the name of the procedure(s) to be replaced. The new version of the procedure must reside in a library or object file known to the SE#OP operator job.

job ! jobs ! j: This parameter specifies the job(s) to be affected by the change. Omission of this parameter will cause all jobs to be affected.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
p procs=(tmrm0a40,tmrm0a41),job=testjob
```


5.3.4 SYSTEM#OP REPERTOIRE

5.3.4.1 Status

The purpose of the status command is to display the status of an entity in the system. The command syntax is as follows.

STATUS ! S name= <ref name> [status= <ref name>]

name ! n: This parameter specifies the name of the entity for which a formatted status report is to be displayed. The LNS type of the name indicates the procedure to be used for data collection and formatting. If there is no predetermined format for a particular type the command degenerates to a "list data" operation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
s tape100
VOLUME 123456 ON TAPE100.
ASSIGNED TO TAPEJOB.
17 RECOVERED READ ERRORS.
```

5.3.4.2 Set

The purpose of the set command is to set various system operational values. The command syntax is as follows.

SET [time= <expr>] [date= <expr>] [tl= <expr>] [rl= <expr>] [bl= <expr>] [bias= (b|i|r[,b|i|r[,b|i|r]])] [status= <ref name>]

time | t: This parameter specifies a string containing the time to which the clock is to be set. It must be entered in the format "hh mm ss".

date | d: This parameter specifies a string containing the date to which the clock is to be set. It must be entered in the format "yy mm dd".

termlimit | tl: This parameter specifies the maximum number of interactive jobs that may be logged in at one time.

remotelimit | rl: This parameter specifies the maximum number of remote batch terminals that may be logged in at one time.

batchlimit | bl: This parameter specifies the maximum number of batch jobs (local or remote) that may be in execution at one time.

bias | b: This parameter specifies the overall biasing of processor dispatching priority among batch (b), interactive(i) and remote batch (r) jobs. The classes are entered in order of priority from left to right.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

NOTE: The selection of parameters for this command is arbitrary for illustration. They will become defined as OS design progresses.

Example:

```
set t='10 10 10',date='75/05/25',bl=6
```

5.3.4.3 Shutdown

The purpose of the shutdown command is to close down a site in an orderly manner. The command syntax is as follows.

SHUTDOWN ! SD [now] [status= <ref name>]

now: This parameter, if quoted, causes the system to close immediately without warm start capability. All activities are canceled. Omission of this parameter will result in a soft shutdown with all activities put into "hold" status with a warm start possible. This parameter should only be used when an urgent shutdown is required.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:
sd

NOTE: A start system command will negate a soft shutdown that is in progress.

5.3.4.4 Hold

The purpose of the hold command is to suspend an activity at the next logical step. The point of suspension is such that a new autoload with a warm start would not affect the activity. For example, a job in the input queue would stay there while an executing job would complete the current job step. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

HOLD ! H name= (<ref name>[,<ref name>]...)!SYSTEM
[status= <ref name>]

names ! name ! n: This parameter specifies the activity or activities to be held. The special name "SYSTEM" represents the set of all activities that may be affected by this command.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:
h (inputq,longjob,sysprt3)

5.3.4.5 Stop

The purpose of the stop command is to suspend an activity immediately. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

STOP ! SP name= (<ref name>[,<ref name>]...)!SYSTEM
[status= <ref name>]

names ! name ! n: This parameter specifies the name of the activity to be stopped. The special name "SYSTEM" represents the set of all activities that may be affected by this command.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
sp (sysrdr1,remote_q)
```


5.3.4.6 Start

The purpose of the start command is to restart an activity previously suspended by a hold or a stop command or to start using a new "onsystem" (see ONSYSTEM command) device. The command syntax is as follows.

START ! ST name= (<ref name>[,<ref name>]...)!SYSTEM
[status= <ref name>]

names ! name ! n: This parameter specifies the name of the activity to be started. The special name "SYSTEM" represents the set of all activities that may be affected by this command.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
start sysrdr2
```

5.3.4.7 Reset

The purpose of the reset command is to reset an activity to its beginning. For example, a listing would reset to the first page and an executing job would return to the login point. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

RESET ! RS name= (<ref name>[,<ref name>]...) [status=<ref name]

names ! name ! n: This parameter specifies the name of the activity to be reset.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
rs snafujob
```


5.3.4.8 Cancel

The purpose of the cancel command is to immediately stop an activity and purge it from the system. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

CANCEL ! C name= (<ref name>[,<ref name>]...) [status=<ref name>]

names ! name ! n: This parameter specifies the name of the activity to be canceled.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
c names=(job4,sysrdr3)
```

5.3.4.9 Alter

The purpose of the alter command is to change the limits or operational parameters of a job. The command syntax is as follows.

```
ALTER ! A job= <ident> [priority= <expr>] [resources=
     (<expr>[,<expr>]...)]    [timelimit=    <expr>]
     [origin=    <expr>]    [destination=    <ref name>]
     [status= <ref name>]
```

job ! j: This parameter specifies the job whose parameters are to be altered.

priority ! p: This parameter specifies the new priority of the job.

resources ! r: This parameter specifies the new resources required by the job.

timelimit ! time ! t: This parameter specifies the new time limit for the job.

origin ! orig ! o: This parameter alters the recorded origin of the job.

destination ! dest ! d: This parameter alters the ultimate disposition of the job's output.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

NOTE: The selection of parameters for this command are arbitrary for illustration. They will become defined as OS design progresses.

Example:

```
alter testjob,priority=17,timelimit=60
```


5.3.5 MASS#OP, TAPE#OP AND BATCH#OP REPERTOIRES

5.3.5.1 Status

The purpose of the status command is to display the status of an entity in the system. The command syntax is as follows.

```
STATUS ! S name= <ref name> [status= <ref name>]
```

name ! n: This parameter specifies the name of the entity for which a formatted status report is to be displayed. The LNS type of the name indicates the procedure to be used for data collection and formatting. If there is no predetermined format for a particular type the command degenerates to a "list data" operation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
s tape100
VOLUME 123456 ON TAPE100.
ASSIGNED TO TAPEJOB.
17 RECOVERED READ ERRORS.
```

5.3.5.2 Hold

The purpose of the hold command is to suspend an activity at the next logical step. The point of suspension is such that a new autoload with a warm start would not affect the activity. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

HOLD ! H name= (<ref name>[,<ref name>]...) [status=<ref name>]

names ! name ! n: This parameter specifies the activity or activities to be held.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
h (inputq,longjob,sysprt3)
```


5.3.5.3 Stop

The purpose of the stop command is to suspend an activity immediately. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

STOP ! SP name= (<ref name>[,<ref name>]...) [status=<ref name>]

names ! name ! n: This parameter specifies the name of the activity to be stopped.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
sp (sysrdr1,remote_q)
```

5.3.5.4 Start

The purpose of the start command is to restart an activity previously suspended by a hold or a stop command or to start using a new "onsystem" device. The command syntax is as follows.

START | ST name= (<ref name>[,<ref name>]...) [status= <ref name>]

names | name | n: This parameter specifies the name of the activity to be started.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
start sysrdr2
```


5.3.5.5 Reset

The purpose of the reset command is to reset an activity to its beginning. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

RESET | RS name= (<ref name>[,<ref name>]...) [status= <ref name>]

names | name | n: This parameter specifies the name of the activity to be reset.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
rs snafujob
```

5.3.5.6 Cancel

The purpose of the cancel command is to immediately stop an activity and purge it from the system. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

CANCEL | C name= (<ref name>[,<ref name>]...) [status= <ref name>]

names | name | n: This parameter specifies the name of the activity to be canceled.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
c names=(job4,sysrdr3)
```


5.3.5.7 Alter

The purpose of the alter command is to change the limits or operational parameters of a job. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

ALTER | A job= <ident> [priority= <expr>] [linelimit= <expr>] [form= <expr>] [dest= <ref name>] [status= <expr>]

job | j: This parameter specifies the job whose parameters are to be altered.

priority | p: This parameter specifies the new priority of the job.

linelimit | lines | l: This parameter specifies the maximum number of lines that the output listing may contain.

destination | dest | d: This parameter specifies a new ultimate destination for the job's output. At this level it may be necessary for the output to be moved to a different queue.

form | f: This parameter specifies the form on which the output is to be printed.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

NOTE: The selection of parameters for this command are arbitrary for illustration. They will become defined as OS design progresses.

Example:

```
alter testjob,priority=17,form='a/p cheques'
```

5.3.5.8 Onsystem

The purpose of the onsystem command is to make available one or more devices to the System Access Manager or File Router. The command syntax is as follows.

ONSYSTEM ! NS device= (<ref name>[,<ref name>]...) queue= (<ref name>[,<ref name>]...) [status= <ref name>]

devices ! device ! dev ! d: This parameter specifies the name(s) of the device(s) to be made available to the system.

queues ! queue ! q: This parameter specifies the name(s) of the queue(s) to be associated with the device(s).

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

NOTE: Several queues may be assocated with one or more output devices but only one queue may be associated with one or more input devices.

Example:

```
onsystem dev=sysrdr3,q=inputq
ns (sysrdr4,aslut200),remote_q
ns (sysprt1,tape100),(outq1,outq2)
```


5.3.5.9 Offsystem

The purpose of the offsystem command is to release one or more devices from system use by System Access Manager or File Router. An "offsystem" device is still online but is available to other jobs. The command syntax is as follows.

OFFSYSTEM ! FS device= (<ref name>[,<ref name>]...) [status= <ref name>]

devices ! device ! dev ! d: This parameter specifies the name(s) of the device(s) to be released.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
fs (tape100,aslut200,sysprt1)
```

5.3.5.10 Backspace

The purpose of the backspace command is to back up an output listing a number of pages. The command syntax is as follows.

BACKSPACE | B device= <ref name> [pages= <expr>] [status= <ref name>]

device | dev | d: This parameter specifies the name of the output device to be backspaced.

pages | p: This parameter specifies the number of pages the listing is to be backspaced. Omission of this parameter will cause a default of one (1) to be assumed.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
backspace device=sysprt4,pages=6
```


5.3.5.11 Forespace

The purpose of the forespace command is to forward space an output listing a number of pages. The command syntax is as follows.

FORESPACE | F device= <ref name> [pages= <expr>] [status= <ref name>]

device | dev | d: This parameter specifies the name of the output device to be forward spaced.

pages | p: This parameter specifies the number of pages to forward space. Omission of this parameter will cause a default of one (1) to be assumed.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
f sysprt2
```

5.3.5.12 Page

The purpose of the page command is to print a specific number of pages of an output listing. The command syntax is as follows.

PAGE | PG device= <ref name> [pages= <expr>] [status= <ref name>]

device | dev | d: This parameter specifies the name of the output device to be paged.

pages | p: This parameter specifies the number of pages to cycle. Omission of this parameter will cause a default of one (1) to be assumed.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
page sysprt2,4
```


5.3.5.13 Online

The purpose of the online command is to add one or more devices to the configuration. The device descriptor(s) must have been previously declared. The command syntax is as follows.

ONLINE | NL device= (<ref name>[,<ref name>]...) [status= <ref name>]

devices | device | dev | d: This parameter specifies the name of the device(s) to be added.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
online disk250
```

5.3.5.14 Offline

The purpose of the offline command is to delete one or more devices from the configuration. The command syntax is as follows.

OFFLINE ! FL device= (<ref name>[,<ref name>]...) [status= <ref name>]

devices ! device ! dev ! d: This parameter specifies the name of the device(s) to be deleted.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
fl (tape100,sysrdr1,tty50)
```


5.3.6 REMOTE#OP REPERTOIRE

5.3.6.1 Status

The purpose of the status command is to display the status of an entity in the system. The command syntax is as follows.

STATUS ! S name= <ref name> [status= <ref name>]

name ! n: This parameter specifies the name of the entity for which a formatted status report is to be displayed. The LNS type of the name indicates the procedure to be used for data collection and formatting. If there is no predetermined format for a particular type the command degenerates to a "list data" operation.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
s remprt4
JOB7 PRINTING ON REMPRT4
```

5.3.6.2 Hold

The purpose of the hold command is to suspend an activity at the next logical step. The point of suspension is such that a new autoload with a warm start would not affect the activity. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

HOLD ! H name= (<ref name>[,<ref name>]...) [status= <ref name>]

names ! name ! n: This parameter specifies the activity or activities to be held.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
h (longjob,remprt3)
```


5.3.6.3 Stop

The purpose of the stop command is to suspend an activity immediately. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

STOP ! SP name= (<ref name>[,<ref name>]...) [status= <ref name>]

names ! name ! n: This parameter specifies the name of the activity to be stopped.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
sp (remrdr1,remote_q)
```

5.3.6.4 Start

The purpose of the start command is to restart an activity previously suspended by a hold or a stop command or to start using a new "onsystem" device. The command syntax is as follows.

START ! ST name= (<ref name>[,<ref name>]...) [status=<ref name>]

names ! name ! n: This parameter specifies the name of the activity to be started.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
start remrdr2
```


5.3.6.5 Reset

The purpose of the reset command is to reset an activity to its beginning. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

RESET ! RS name= (<ref name>[,<ref name>]...) [status=<ref name>]

names ! name ! n: This parameter specifies the name of the activity to be reset.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
rs snafulist
```

5.3.6.6 Cancel

The purpose of the cancel command is to immediately stop an activity and purge it from the system. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

CANCEL ! C name= (<ref name>[,<ref name>]...) [status= <ref name>]

names ! name ! n: This parameter specifies the name of the activity to be canceled.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
c names=(job4,sysrdr3)
```


5.3.6.7 Alter

The purpose of the alter command is to change the limits or operational parameters of a job. Refer to "5.4 Command Summary" for a description of the activities that may be referenced by this command at this level of privilege. The command syntax is as follows.

ALTER ! A job= <ident> [priority= <expr>] [linelimit= <expr>] [form= <expr>] [dest= <ref name>] [status= <expr>]

job ! j: This parameter specifies the job whose parameters are to be altered.

priority ! p: This parameter specifies the new priority of the job.

linelimit ! lines ! l: This parameter specifies the maximum number of lines that the output listing may contain.

destination ! dest ! d: This parameter specifies a new ultimate destination for the job's output. At this level it may be necessary for the output to be moved to a different queue.

form ! f: This parameter specifies the form on which the output is to be printed.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

NOTE: The selection of parameters for this command are arbitrary for illustration. They will become defined as OS design progresses.

Example:

```
alter testjob,dest='acctng',form='a/p cheques'
```

5.3.6.8 Backspace

The purpose of the backspace command is to back up an output listing a number of pages. The command syntax is as follows.

BACKSPACE ! B device= <ref name> [pages= <expr>] [status= <ref name>]

device ! dev ! d: This parameter specifies the name of the output device to be backspaced.

pages ! p: This parameter specifies the number of pages the listing is to be backspaced. Omission of this parameter will cause a default of one (1) to be assumed.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
backspace device=remprt4,pages=6
```


5.3.6.9 Forespace

The purpose of the forespace command is to forward space an output listing a number of pages. The command syntax is as follows.

FORESPACE ! F device= <ref name> [pages= <expr>] [status= <ref name>]

device ! dev ! d: This parameter specifies the name of the output device to be forward spaced.

pages ! p: This parameter specifies the number of pages to forward space. Omission of this parameter will cause a default of one (1) to be assumed.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
f remprt2
```

5.3.6.10 Page

The purpose of the page command is to print a specific number of pages of an output listing. The command syntax is as follows.

PAGE | PG device= <ref name> [pages= <expr>] [status= <ref name>]

device | dev | d: This parameter specifies the name of the output device to be paged.

pages | p: This parameter specifies the number of pages to cycle. Omission of this parameter will cause a default of one (1) to be assumed.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
page remprt2,2
```


5.3.6.11 Online

The purpose of the online command is to add one or more devices to the remote configuation. The device descriptor(s) must have been previously declared. The command syntax is as follows.

ONLINE | NL device= (<ref name>[,<ref name>]...) [status= <ref name>]

devices | device | dev | d: This parameter specifies the name of the device(s) to be added.

status | s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:

```
online remrdr2
```

5.3.6.12 Offline

The purpose of the offline command is to delete one or more devices from the remote configuration. The command syntax is as follows.

OFFLINE ! FL device= (<ref name>[,<ref name>]...)
[status= <ref name>]

devices ! device ! dev ! d: This parameter specifies the name of the device(s) to be deleted.

status ! s: The status parameter specifies a variable into which the status is to be returned. Omission of the status parameter will cause the SCL error handler to be invoked upon the occurrence of an error condition.

Example:
fl (remrdr1,tty50)

5.4 COMMAND SUMMARY

The following table illustrates the availability of commands to different levels of operators and the activities that may be affected by them. The following abbreviations are used in the table.

ok	no restrictions
na	not available
-->	included via lower level repertoire
system	entire system
q	input or output queue
opq	output queue
ropq	remote batch output queue
job	job
rjob	remote origin job
opqjob	job in output queue or function
ropqjob	job in remote batch output queue or function
dev	device
sysdev	"onsystem" device
rdev	remote batch terminal device
rsysdev	remote batch terminal "onsystem" device

To use the table, locate the intersection of the command name and the logical operator job name. The activities that may be referenced by the command when used by the particular logical operator are all those to the right of the intersection in the row for that command. If an arrow (-->) is encountered, follow it.

5.0 OPERATOR INTERFACE
5.4 COMMAND SUMMARY

	CE#OP	SE#OP	SYSTEM#OP	MASS#OP TAPE#OP BATCH#OP	REMOTE#OP
reply	-->	-->	-->	ok	rjob
inform	-->	-->	-->	ok	rjob
status	-->	-->	-->	ok	ropq rjob rdev
seize	ok	na	na	na	na
unseize	ok	na	na	na	na
memory	-->	ok	na	na	na
patch	-->	ok	na	na	na
set	-->	-->	ok	na	na
shutdown	-->	-->	system	na	na
hold	-->	-->	system q job	opq opqjob sysdev	ropqjob rsysdev
stop	-->	-->	system q job	opq opqjob dev	ropqjob rdev
start	-->	-->	system q job	opq opqjob dev	ropqjob rdev
reset	-->	-->	job	opqjob sysdev	ropqjob rsysdev
cancel	-->	-->	q job	opq opqjob sysdev	ropqjob rsysdev
alter	-->	-->	job	opqjob	ropqjob

continued...

5.0 OPERATOR INTERFACE
5.4 COMMAND SUMMARY

	CE#OP	SE#OP	SYSTEM#OP	MASS#OP TAPE#OP BATCH#OP	REMOTE#OP
onsystem	-->	-->	-->	dev	rdev
offsystem	-->	-->	-->	dev	rdev
backspace	-->	-->	-->	sysdev	rsysdev
forespace	-->	-->	-->	sysdev	rsysdev
page	-->	-->	-->	sysdev	rsysdev
online	-->	-->	-->	dev	rdev
offline	-->	-->	-->	dev	rdev

6.0 ADDITION OF NEW COMMANDS

The addition of new commands to OCS is the same as for any other instance of SCL and is described in the Command Writers Guide. Consideration must be given to the various interactions involved for some commands that affect other functional areas of the system.

7.0 OPERATING SYSTEM DEPENDENCIES

The OCS system described is highly dependent on certain features and co-operative action being present in the operating system. These fall into two major categories.

First, all OCS commands require that referenced parameters (queues, jobs, devices, etc.) may be located via LNS in the global segment or in the local segment of the least privileged user of a particular command. This access may be achieved by the data being in LNS, a pointer or pointers to the data in LNS or the descriptor of a procedure that is capable of locating the data residing in the proper LNS segment. Note that with the exception of the status command, write access is often required.

Second, each area of the system must ensure that the set of primitives required to complete an OCS command is made available. In many cases, a command may require complementary operations from several system areas.

The OCS system implementation is a "top down" definition. An attempt has been made to describe an efficient and practical operating facility without immediate regard for the capabilities of the Operating System. Conversely, the system repertoire is "bottom up" and is an externalization of defined Operating System capabilities with simplification and combination added at the command level.

Particular note should be made of the "status" command. This command procedure is designed for maximum flexibility and may function in three ways. The name parameter is located in LNS and a data type dependent procedure is called to format the status display. If the data type has no associated procedure, the contents of the named variable are listed according to standard SCL conventions. An exception to this process occurs when the LNS data type is "procedure". In this case, the named procedure is invoked to produce the status report. The result of the above technique is that an LNS descriptor may be listed in "status" form, a status report may be maintained in global or operator LNS as an array of strings

or special status requests may be issued to task services for access to information not in LNS.

8.0 ERROR CONDITIONS

To be supplied.

8.1 DEFINITION OF CODES

To be supplied.