Burroughs

B 2000/B 3000/ B 4000 Series System Security

INSTALLATION AND OPERATION GUIDE

(RELEVANT TO ASR 6.6 SYSTEM SOFTWARE RELEASE)

This Manual Replaces All Previous Editions of Form 1127362

Printed in U.S. America November 1982 1127362

LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii	Original
iii	Original
iv	Blank
v thru ix	Original
x	Blank
1-1 thru 1-4	Original
2-1 thru 2-2	Original
3-1 thru 3-11	Original
3-12	Blank
4-1 thru 4-6	Original
5-1 thru 5-7	Original
5-8	Blank
6-1 thru 6-3	Original
6-4	Blank
1 thru 2	Original

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

LIST OF TABLES

PREFACE

This manual describes the security features of the B 2000/B 3000/B 4000 Series Systems.

Disk File Security is an MCP option which can be invoked to control programmatic and operator access to permanent disk and diskpack files.

ORGANIZATION OF THE MANUAL

This manual is divided into the following sections.

Section 1: Introduction
This section introduces the security attributes, defines terms, and describes the railroad syntax structures used throughout this manual.

Section 2: Security Capabilities
This section defines the capabilities and functions of the security system. It is necessary to understand the information in this section before implementing Security.

Section 3: USERHO and USERLS
This section describes the syntax for input to the USERHO program. USERHO is used to specify the usercodes and their attributes to the system.

Section 4: Operating Procedures
This section discusses the operational and programmatic ramifications of a Secured system.

Section 5: Utility Programs
This section describes the MCP utility programs which interface with Security.

Section 6: Language Requirements
This section gives the syntax necessary to invoke security attributes for COBOL, FORTRAN, PASCAL, and CANDE.

RECOMMENDED BACKGROUND

The user of this manual should have a functional understanding of the Medium Systems Master Control Program (MCP). Although an interface exists between Security and CANDE, familiarity with CANDE is unnecessary unless the Secured system also runs CANDE.

RELATED PUBLICATIONS

The following manuals contain additional material on topics referred to in this manual:

B 2000/B 3000/B 4000 Series MCP System Software Operation Guide, Vols 1 & 2, form nos. 1127529 and 1127321.

B 2000/B 3000/B 4000 Series CANDE Installation and Operations Guide, form no. 1108834.

B 2000/B 3000/B 4000 Series MCP System Software Functional Description Manual, form no. 1090685.

B 2000/B 3000/B 4000 Series COBOL ANSI-74 Language Manual, form no. 1090735.

SECTION 1
INTRODUCTION

The System Security feature of the MCP allows a site to protect data files and programs against unauthorized access. In addition to controlling access to information stored in the system, the Security system can also restrict access to the system itself. In other circumstances, system access can be allowed while controlling each user's capabilities on the system.

DEFINITIONS

Certain terms used in this manual are defined in the following paragraphs.

Usercode and Password

These terms refer to information which is entered into the system when attempting to gain access. Usercode and password each consist of a maximum of 10 alphanumeric characters. Passwords are dependent on usercode, although a usercode may or may not have a password associated with it. Any usercode/password combination can be presented to the MCP for verification but only the combinations that have been defined through the USERHO program will be allowed access to the system.

Access code

An access code consists of a usercode/password/ charge number combination. This information is required for log-in. After the log-in information has been verified, the usercode portion is used to identify the user and to determine his capabilities. For example, an access code must be supplied in order to execute a program. Once the program is executing, only the usercode from the access code is used to determine if the program can access the files that it needs. The password and charge number have no effect. Password information is not retained once the log-in is accomplished. The charge number is retained for logging.

SECURITYTYPE

SECURITYTYPE indicates the type of security a file has. Valid values are PRIVATE, PUBLIC, or GUARDED. PRIVATE specifies that only the creator of the file can access it, while PUBLIC implies that the file is accessible to any usercode. GUARDED means the access privileges for this file are specified in an external file called a guard file.

Guard file

A guard file is a disk or diskpack file that contains information used to determine the access privileges of particular usercodes. Guard files are created using the utility program GUARD.

SECURITYUSE

SECURITYUSE specifies how a file may be OPENed. Options are IN (read only), OUT (write only), IO (read or write) or SECURED. If the value is SECURED, the file is a program and can only be executed.

RAILROAD SYNTAX

Railroad syntax diagrams show how syntactically valid statements can be constructed. Traversing a railroad syntax diagram syntax diagram from left to right, or in the direction of the arrowheads, and adhering to the limits illustrated by bridges produces a syntactically valid statement.

Continuation from one line of a diagram to another is represented by a right arrow (>) appearing at the end of the current line and at the beginning of the next line. Railroad syntax diagrams are terminated by a vertical bar (|).

Items contained in broken brackets (< >) are syntactic variables which are further defined in the manual, or which require the user to supply the requested information.

Upper-case items must appear literally. When abbreviations are permitted, the minimum abbreviation is underlined.

The general format of a railroad syntax diagram is shown in figure 1-1.

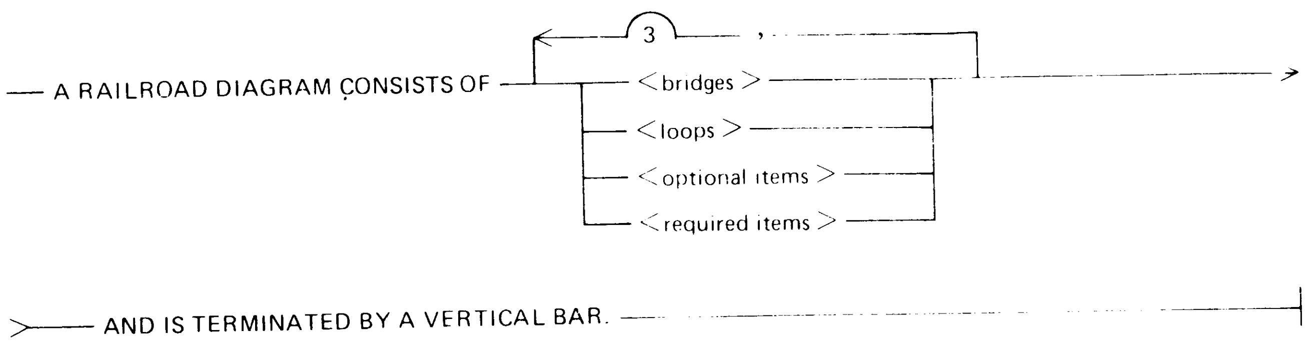

A RAILROAD DIAGRAM CONSISTS OF
3
,
<bridges>
<loops>
<optional items>
<required items>
AND IS TERMINATED BY A VERTICAL BAR.

Figure 1-1. Railroad Syntax Diagram

The following syntactically valid statements can be constructed from figure 1-1:

A RAILROAD SYNTAX DIAGRAM CONSISTS OF <bridges> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD SYNTAX DIAGRAM CONSISTS OF <optional items> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD SYNTAX DIAGRAM CONSISTS OF <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD SYNTAX DIAGRAM CONSISTS OF <optional items>, <required items>, <bridges>, <loops>, AND IS TERMINATED BY A VERTICAL BAR.

<required items>

In the railroad syntax diagram shown in figure 1-2 no alternate path exists for required items or required punctuation.

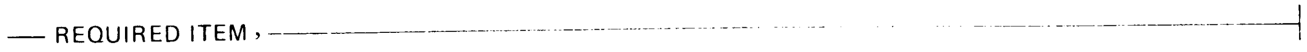


Figure 1-2. <required items> Railroad Syntax

<optional items>

Items shown as a vertical list indicate that the user must make a choice of the items specified. An empty path through the list allows the <optional item> to be absent as shown in figure 1-3.

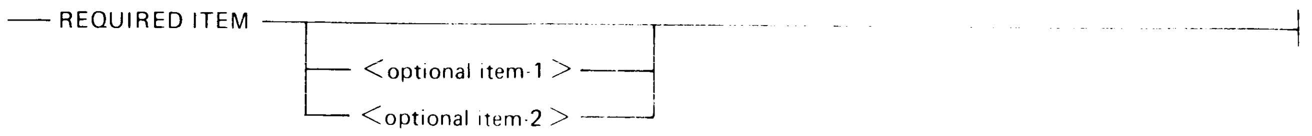


Figure 1-3. <optional items> Railroad Syntax

Three statements can be constructed from this diagram:

REQUIRED ITEM
REQUIRED ITEM <optional item-1>
REQUIRED ITEM <optional item-2>

<loops>

A loop is a recurrent path through a railroad syntax diagram and has the general format shown in figure 1-4. A specific example is shown in figure 1-5.

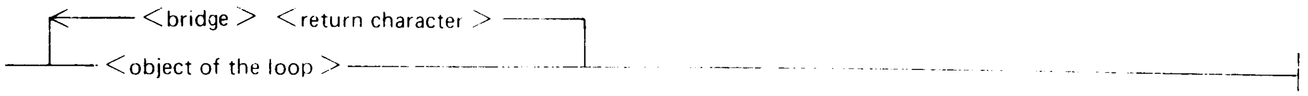


Figure 1-4. <loops> Railroad Syntax (General Format)

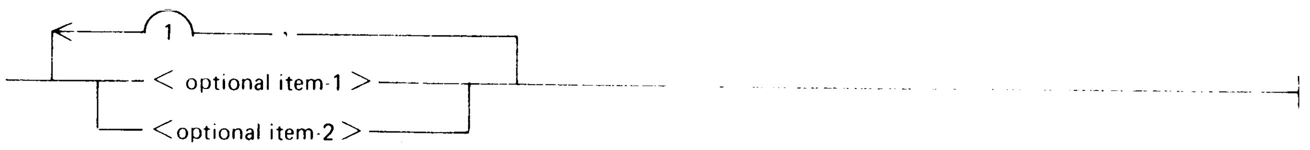


Figure 1-5. <loops> Railroad Syntax (Specific Example)

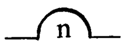
The following examples show some of the statements which can be constructed from figure 1-5:

<optional item-1>
<optional item-1>, <optional item-1>
<optional item-2>, <optional item-1>

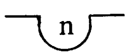
A <loop> must be traversed in the direction of the arrow, and the limits specified by the bridges cannot be exceeded.

<bridges>

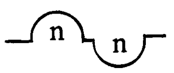
There are three forms of <bridges> as shown in figure 1-6:



n is an integer that specifies the maximum number of times the path can be traversed.



n is an integer that specifies the minimum number of times the path must be traversed.



n is an integer that specifies the exact number of times the path must be traversed.

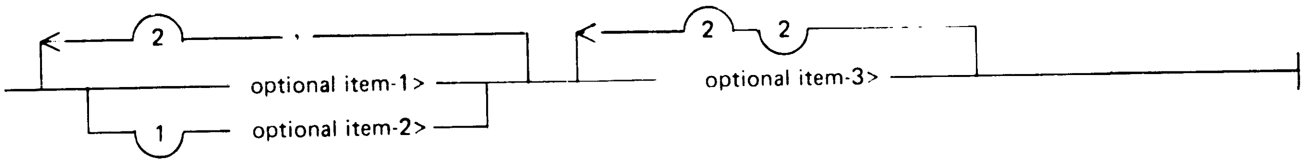


Figure 1-6. <bridge> Railroad Syntax

The first loop can be traversed a maximum of two times; however, the path for <optional item-2> must be traversed at least one time. The loop for <optional item-3> must be traversed two times.

Several statements can be constructed from figure 1-6. Some of these statements are as follows:

<optional item-1>, <optional item-2>, <optional item-3>, <optional item-3>,<optional item-3>

<optional item-2>, <optional item-2>, <optional item-1>, <optional item-3>, <optional item-3>, <optional item-3>

<optional item-2>, <optional item-3>, <optional item-3>, <optional item-3>

SECTION 2
SECURITY CAPABILITIES

GENERAL

Security is not required to run any Medium Systems installation. On a system which is not using Security any user can perform any function in the system. There are no restrictions on the type of keyboard commands that can be entered, the programs that can be executed, nor the files that can be programmatically accessed. Without Security, complete control of the system is available to every user.

However, such unlimited access may not be always be desirable. While some individuals in an organization require such access, other personnel need no more than limited access to the operating system.

A SECURED SYSTEM

The basic attributes of a Secured system are described in the remainder of this section.

System Access

Under Security, each user is required to submit identification to the system before any capabilities are granted by the MCP. This identification is the log-in message (LI) or a control statement such as USER or BEGINUSER. No system access is allowed until a valid user access code is given to the MCP.

The definition of the access code includes specifying a SPO level to be associated with the usercode. The SPO level determines what keyboard input commands this user can use. If the user is logged-in at the OCS, keyboard inputs are checked against the usercode level to determine whether such input is acceptable. Furthermore, if this user executes a program from the OCS, his usercode will be associated with the program. Whether the program is able to access files depends on the Security attached to the required files.

The System Software Operation Guide lists the valid keyboard input commands and their associated levels. A higher level gives greater capability.

A unit remains logged-in until it is logged-off, and this log-in is associated with all inputs from the unit, whether program executes or operator inquiries. The log-off message (LO) is intended to prevent the accidental, unauthorized use of a usercode.

File Access

All files created under Security have an attribute that designates how a file is secured:

1. If a file is PUBLIC, any user can access it.
2. If PRIVATE, only a user with a matching usercode can access it.
3. If GUARDED, access rights are determined by a guard file.

The mode of access can also be controlled. A file can be designated to be used IN, OUT, or IO; or it can be declared a program and permitted only to be executed.

A Security violation occurs when a program attempts to access a file under a non-matching usercode or an improper access mode. These violations cause the program to be aborted and an appropriate Run Log entry made. An ARMed program will be reinstated at its ARM branch.

If a file is GUARDed, a guard file is interrogated to determine a user's access to the file. Guard files can selectively allow usercodes to access files. Access to PRIVATE files is based solely on matching usercodes. The guard file is flexible enough to specify that a file may be accessed by multiple usercodes with different access capabilities; for example, the guard file may specify that usercode A can access file X input only, while usercode B can access file X input/output.

Guard files may be located on disk or diskpack.

Library Maintenance

Since files can be accessed only by programs running under matching usercodes, backing-up files to tape using system utilities could result in a vast number of tapes (one for each usercode). To avoid this, a usercode can be given library maintenance privileges. This privilege allows the usercode to load and dump files created under other usercodes, as well as change file names and remove files.

Privileged User

Certain usercodes may be designated such that certain users are not subject to Security's checks and restrictions. Such a user is known as a privileged user. A privileged user can access any file in any manner and perform any library maintenance function.

Charge Numbers

Under Security, charge number validation is possible. When usercodes are set up, the access codes (<usercode>/<password>/<charge no.>) input to the USERHO program determine what usercodes will be valid. Specifying a charge number in the access code will require that specific charge number be used with the usercode and password.

ZIPs

Programs initiated with the ZIP mechanism assume the usercode characteristics of the originator, unless the ZIPed text includes a USER or BEGINUSER statement to designate another usercode. Usercode and charge number are carried forward to created programs.

Control or SPO text which is ZIPed undergoes the same checking that is performed on the text when it is entered at the OCS. The SPO level must be appropriate as must certain attributes of the usercode.

Device Alternates

Backup files created under a usercode take on the attributes of that usercode, unless overridden by a File Equate statement. For example, if the creator of a printer backup file has a usercode which specifies PRIVATE files as default, then the printer backup file is a private file and can be accessed only by the correct usercode.

Pseudo decks created under a usercode do not carry the attributes of that usercode. A pseudo deck is assigned the usercode FMCP01 when it is activated (see section 5), unless the pseudo deck contains USER and BEGINUSER statements designating another usercode.

SECTION 3
USERHO AND USERLS

GENERAL

USERHO is the program used to create and update the USERFL file. USERFL is a NO LIBMAINT MCP file which contains usercode/password/charge number information that is used to determine what access privileges to the system, to terminals, and to files will be granted. The input to USERHO describes what input devices may access the system, the valid access codes, and functional capabilities. USERHO program

The USERLS program used to list the contents of the USERFL file will be discussed following later in this section.

USERHO

Input to USERHO can come from cards (file-id = USER) or from an EDITOR file when executed with SW 8 = 1, file-id = EDITOR. Card data is located in columns 1-72; columns 73-80 are used for sequence numbers. Comments can be used on any card except the dollar option card. Comments must begin with a period. All input statements to USERHO except the $-card are terminated by semicolons (;).

The following commands are used to execute USERHO:

```
? EX USERHO
? DATA USER
 [< $ card >]
 [<CLEAR card>]
 [<MEDIA section>]
 [<FUNCTION section>]
 [<ACCESSCODES section>]
? END
```

The Dollar Card

The $ card is used to specify optional operational parameters to USERHO. The $-sign can appear in any column, and the options can be listed in any order.

The following options can appear on the $ card.

LIST
: Specifies that a single-spaced listing of the input is desired. LIST is the default when no $ card is specified in the input deck.

NOLIST
: Specifies that no listing is desired. This option is overridden if syntax errors are detected.

SYNTAX
: Process the input and check for syntax errors only; do not perform any USERFL updates.

LIBRARY
Process the input and update USERFL if no errors are found. This option is default if no $ card is present.

CLEAR Card

The syntax of the CLEAR card is given in figure 3-1.

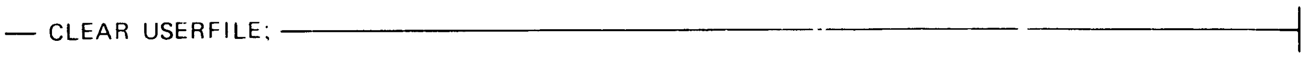


Figure 3-1. CLEAR Statement Syntax

This statement is used to clear the current contents of the USERFL before any updates are performed. All information is cleared except the information associated with the usercode under which USERHO is running.

A null schedule and a null mix (except for this execution of USERHO), on all processors (if SHARED), is required for the CLEAR statement to function.

MEDIA Section

The MEDIA section describes groups of input devices from which users can log-in to the system. Media specifications are optional. Later in the input, usercodes will be specified which will be associated with the media described in this section. If a usercode is not associated with any media specification, it is valid from all input media.

The format of the MEDIA section is shown in figures 3-2 and 3-3.

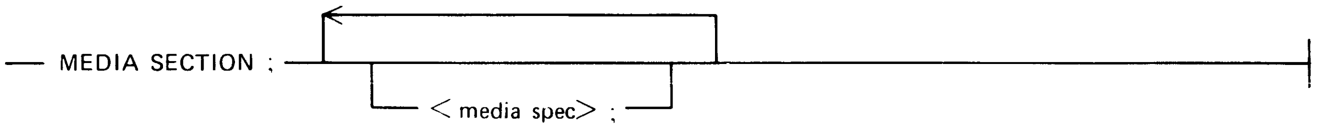


Figure 3-2. MEDIA Section Syntax

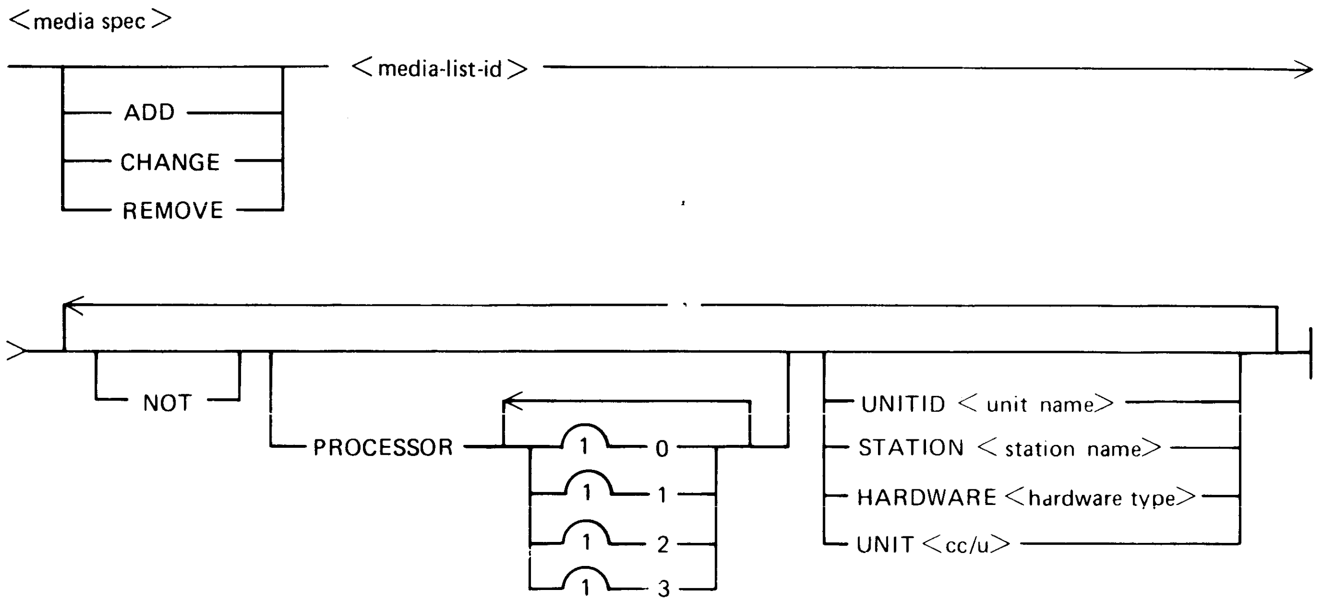


Figure 3-3. <media spec> Syntax

Semantics

If no action code is specified, ADD is assumed.

For REMOVE, only < media-list-id > should be entered.

A CHANGE is an implicit REMOVE of all parameters followed by an ADD of the CHANGE parameters.

< media-list-id >
This is a 1-to 10-character identifier consisting of upper/lower case letters and/or numbers. The first character must be alphabetic. Within the media specifications, a <media-list-id> must be unique. The <media-list-id> is used in the ACCESSCODES section to specify the devices which may be logged-in under a particular access code.

NOT
The word NOT defines which devices may not log-in to the system. If all clauses specify NOT, the user may log-in from any input-capable device which is not specified. If none of the clauses specifies NOT, the user will be able to log-in only from the devices specified. A combination of NOT and non-NOT clauses within the same media specification has the same result as having only the clauses without NOT.

PROCESSOR
The PROCESSOR clause indicates that the following media description applies to all specified processors. Omitting PROCESSOR means that the declaration applies only to the processor on which USERHO is executed.

UNITID
This clause is used to select a datacomm line. The <unit name> is the adapter-id specified on the UNIT card in the system environment deck.

STATION
This clause is used to select a CANDE station attached to a DCP. The <station name> must be the same as that declared in the NDL compilation.

HARDWARE
This clause selects one or more devices by hardware type. Valid hardware types are CRD, SPO, OCS, TC4, RJE, TWX, and TYP.

UNIT
This clause selects a device by physical location. The channel number must be in the range 0 – 77 and the unit number must be in the range 0 – 15.

For the device specifications in the media clauses, the specified devices need not be physically present nor declared to the system.

Examples:

```
ADD YALLL PROCESSOR 0 1 2 HARDWARE OCS,
          PROCESSOR 0 1 2 HARDWARE CRD,
          PROCESSOR 0     STATION SYSTEM;
ADD OTHERS NOT HARDWARE OCS, NOT STATION SYSTEM;
CHANGE ONESYS PROCESSOR 2; .Changes ONESYS to reflect
                           .only processor 2
REMOVE EXTRAPROC ;
```

FUNCTION Section

This section is used to describe the functions that a user or a group of users may perform. The syntax of the FUNCTION section is shown in figures 3-4 and 3-5.

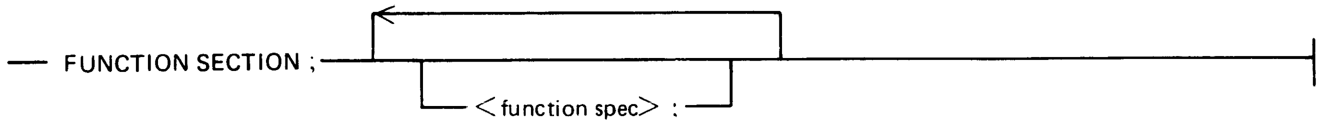


Figure 3-4. FUNCTION Section Syntax

If no action is specified, ADD is assumed.

For REMOVE, the < function clauses> must be omitted.

A CHANGE is an implicit REMOVE followed by an ADD of the CHANGE parameters.

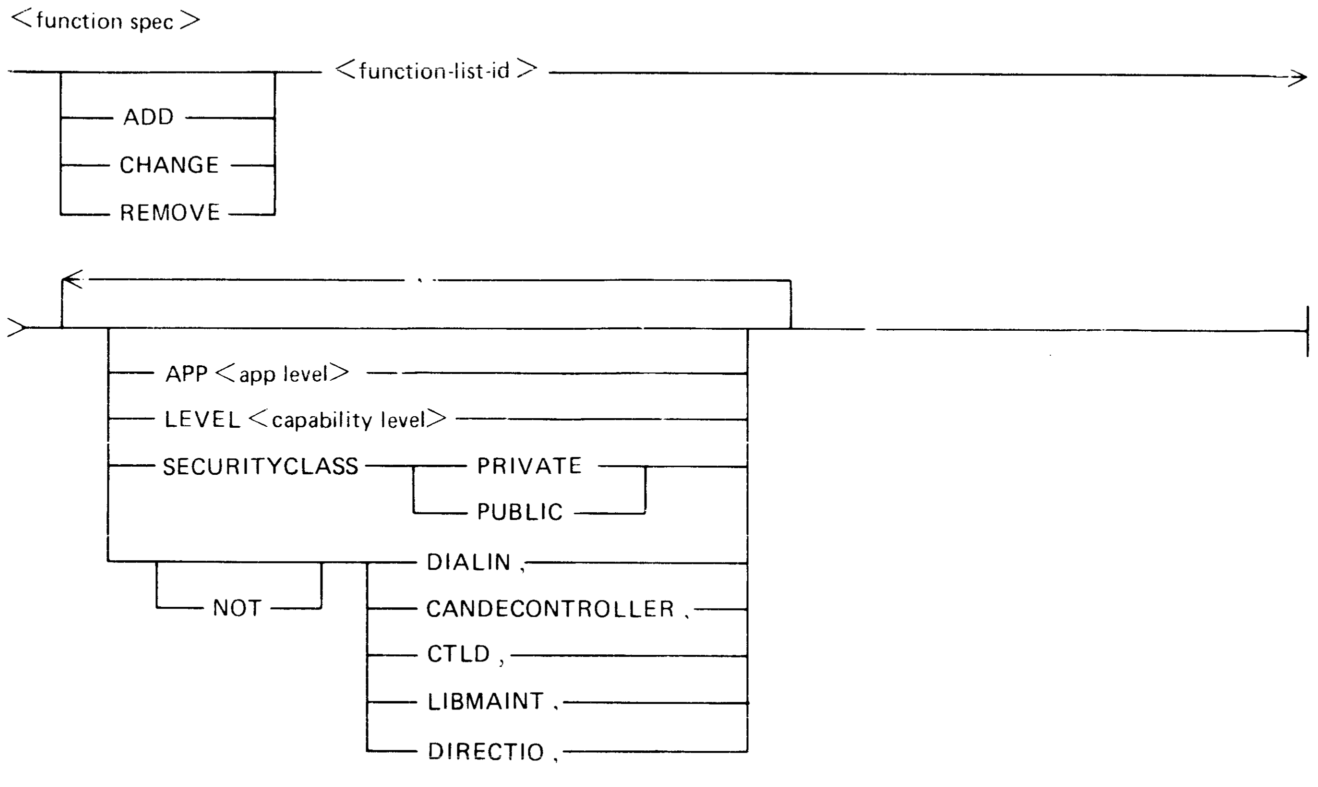


Figure 3-5. <function spec> Syntax

Semantics

< function-list-id >

This is a 1-to 10-character identifier consisting of upper/lower case letters and/or numbers. The first character must be alphabetic. Within function specifications, a <function-list-id> must be unique. The <function-list-id> is used in the ACCESSCODES section to associate permissable functions with an access code.

APP

This specifies the default application mode for a user when logging-in through a CANDE terminal capable of the specified <app mode>. Values for <app mode> range from 1 to 6. If APP is not specified, the default value is @F@, indicating that no APP clause was specified; the mode assigned to the terminal is the CANDE default APP mode.

LEVEL

This specifies the SPO and control card capability level for a user and corresponds to the SPO level for OCS and remote SPO devices. Because this specification applies to a user, it is "portable" in that the capability level applies wherever the user logs in to the system. The levels for OCS and remote SPOs apply only to the device. The actual level assigned to a user will be the lower level of that specified in the <function clause> and the terminal from which he is operating.

Valid levels range from 0 – 9. The default level is zero. While LEVEL and FUNCTION specifications themselves are optional, at least one function specification must be made to assign a non-zero capability level in order to operate the system. Central site operators should be assigned usercodes with level 9 capability, while a level of 5 is usually sufficient for RJE and TSM users.

Refer to the OCS UNIT card and Data Communications keyboard messages in the MCP System Software Operation Guide, vol. 1, for an explanation of levels.

SECURITYCLASS
This specifies a default security class for all disk and diskpack files created by a user, which do not have a security class otherwise associated with them (compiler work files excepted). The security class can be set up through FIB fields, COBOL file attributes, or label equation. The default SECURITYCLASS is PRIVATE.

DIALIN
This specifies that a user may log-in at a CANDE terminal connected to a dialed line. DIALIN provides an additional level of security in that only specific users can access the system from the telephone system. The default value is FALSE, or NOT DIALIN.

CANDECONTROLLER
This gives the user access to certain privileged functions in the CANDE system. CANDECONTROLLER is the same boolean as PRIVILEGED or PRIV in USERTS (in CANDE). Default is FALSE, or NOT CANDECONTROLLER.

CTLD
This is used by CANDE and EDITOR to determine whether the user has the capability of COPY TO CTLD to create pseudo-decks. Default is FALSE.

LIBMAINT
This is used to permit a user limited access to other users' files regardless of their security attributes. LIBMAINT allows the system operator to LOAD, DUMP, REMOVE, or CHANGE any file on the system. Because this feature allows a user to remove and dump files which do not belong to his usercode, care should be exercised in allocating this capability. The capabilities of file security could be compromised. The default value is FALSE.

DIRECTIO
This allows a user to have direct I/O access to the various devices on the system. Because the use of direct I/O can compromise the security system, the assignment of DIRECTIO should be seriously considered. The default is NOT DIRECTIO, or FALSE.

ACCESSCODES Section

The ACCESSCODES section describes the codes which can be used to gain access to the system, attributes associated with the access codes, codes, and linkages from the access codes to the MEDIA and FUNCTION descriptions.

The syntax of the ACCESSCODES section is shown in figures 3-6 to 3-10.

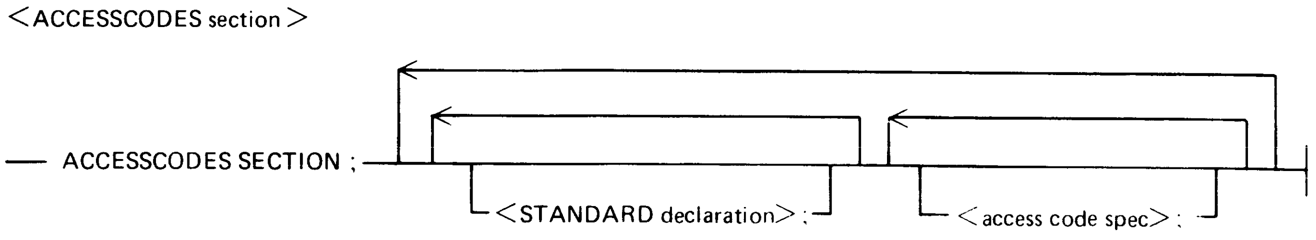


Figure 3-6. ACCESSCODES Section Syntax

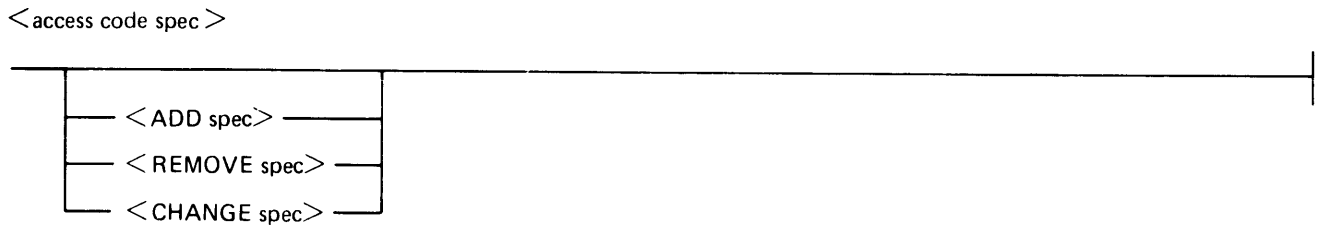


Figure 3-7. <access code spec> Syntax

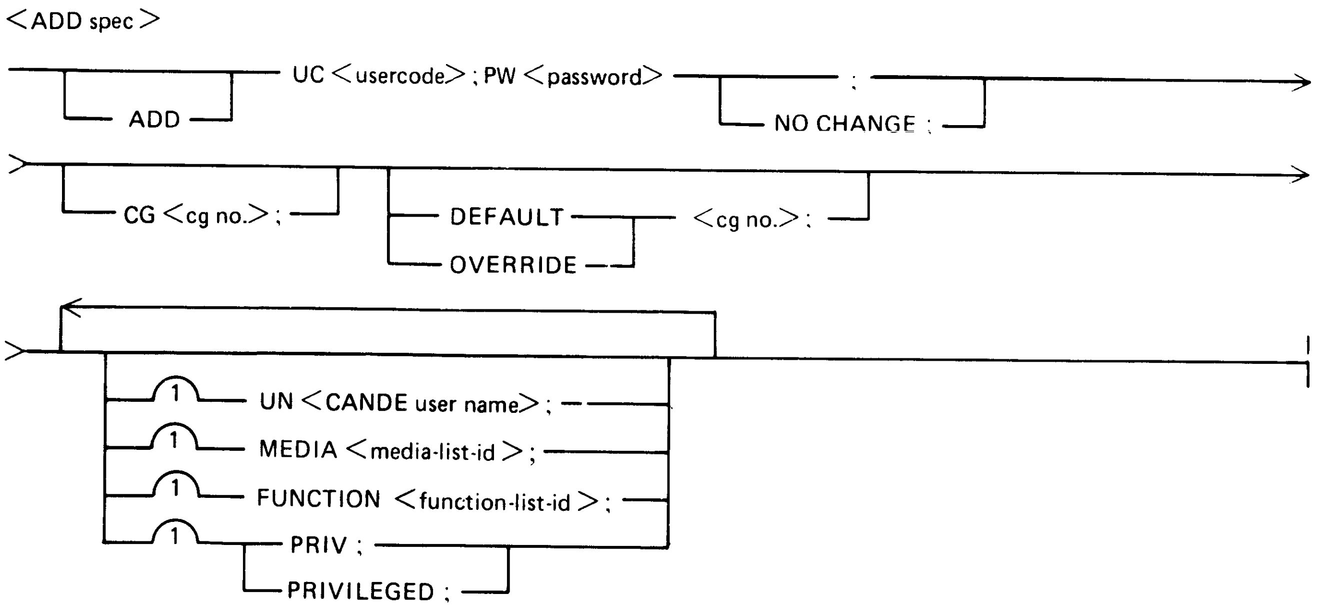


Figure 3-8. <ADD spec> Syntax

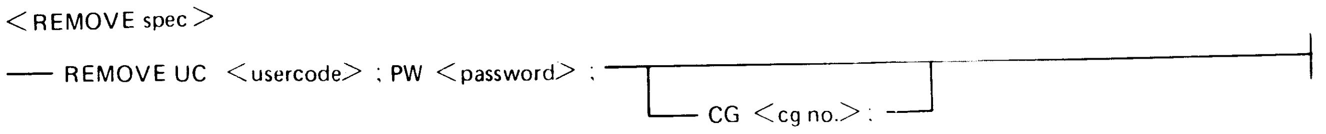


Figure 3-9. <REMOVE spec> Syntax

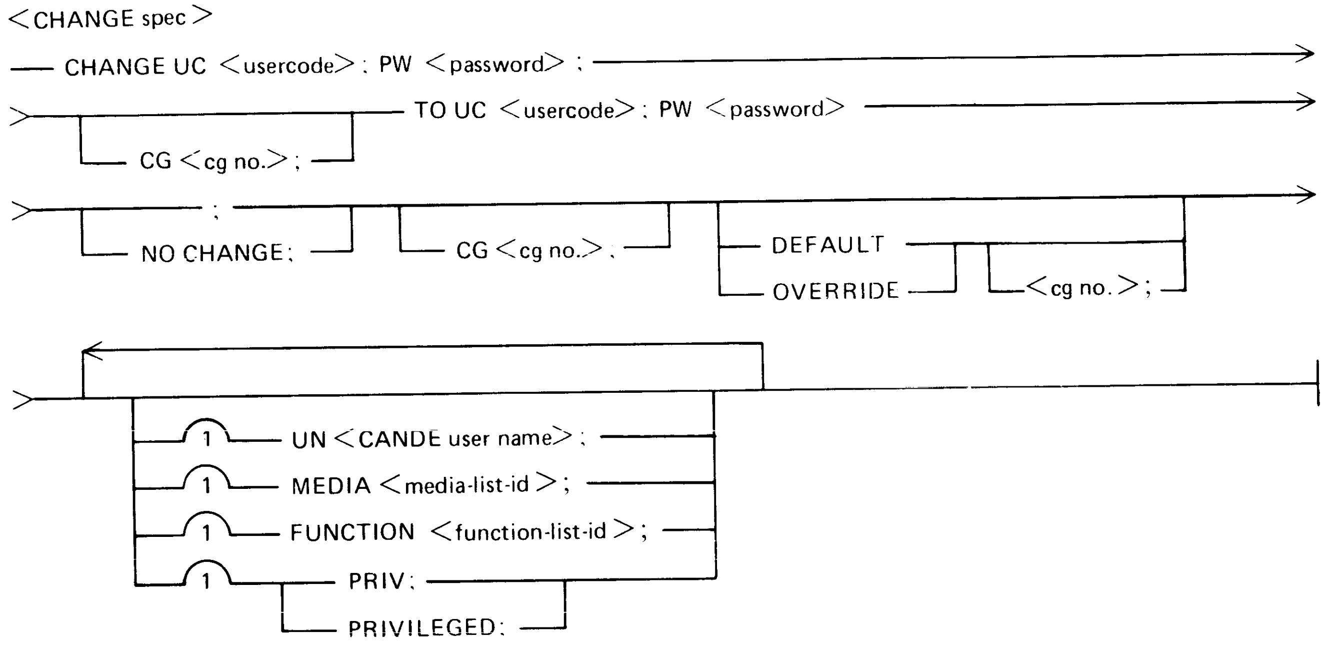


Figure 3-10. <CHANGE spec> Syntax

Semantics

UC,PW,CG

This combination (usercode, password, charge number) comprises the access code. Each of the syntactic items (<usercode>, <password>, and <cg no.>) can be specified in one of four ways:

1. N/A (not allowed). If a field is specified as N/A, no value may be supplied for the field when a log-in is attempted. Supplying a value causes an access code mismatch.
2. N/R (not required). If a field is specified as N/R, any value which is syntactically correct for the field will be accepted and the supplied value will be stored for future interrogation. N/R is assumed for CG if the CG parameter is omitted.
3. A specific value. For a usercode or password, this value is a letter followed by 1-9 upper-and/or lowercase letters and/or numbers, or a number followed by 1-9 numbers; it is 1-6 digits for a charge number. The corresponding field in the log-in attempt must match the specified field exactly; otherwise, an access code mismatch occurs. For CANDE usercodes, the specific value must be numeric; only the first four digits are used.

NOTE

For CANDE users, because of the manner in which usercodes are appended to EDITOR files, it is strongly suggested that an installation be selective in assigning usercodes. File maintenance is considerably simpler if usercodes which result in printable characters are used, making the files easy to identify on OCS and printer displays with lower-case capability.

In general, any digit pair which results in a meaningful graphic character (such as the lower-case letters "a" through "i" (81 through 89), "j" through "r" (91 through 99), and symbols such as "&" (50) and "-" (60)) is preferable to a pair which is not representable or which has a particular meaning, such as 03 in datacom. In particular, the digit pairs 40 and below should be avoided.

4. A masked value is similar to a specific value except that = signs may be used for ignored positions. Those characters which are not masked must match exactly those supplied in the log in text. Where a position is masked, any syntactically valid character may be supplied. The text supplied in the log-in will be stored for future use. Masking an entire field is the same as specifying N/R. Trailing masking is not implied by one equal sign. If trailing masking is desired, each position must be specified.

NO CHANGE

This parameter specifies that a password may not be changed. Passwords may be changed unless NO CHANGE is specified.

The following parameters may appear in any sequence following usercode, password, and charge number.

DEFAULT

This clause is used when a charge number is to be associated with all jobs although CG N/R has been specified. If the user supplies a charge number, it is used. If a charge number is not supplied, the DEFAULT number will be used. The default number must be a specific 1-to 6-digit number. The DEFAULT entry may not be masked, N/A, nor N/R. DEFAULT may not be used with OVERRIDE.

OVERRIDE

This is used when a predetermined charge number is desired in place of one which the user might enter. Validation is performed on the user-supplied charge number. If valid, the supplied charge number is replaced by the OVERRIDE value. The OVERRIDE entry may not be masked, N/A, or N/R. OVERRIDE and DEFAULT may not appear in the same specification.

UN

This parameter supplies a 1-to 10-character alphanumeric-id to be used by the CANDE system as an external user-id. If this parameter is omitted, the usercode will be used as the id by default. If this parameter is used, the usercode may neither be N/A nor N/R.

MEDIA

This is used to associate a particular media list with an access code to restrict a user to certain terminals. The <media-list-id> must match one of those declared in the MEDIA section. An access code which does not have a MEDIA clause may be used from any peripheral on the system capable of logging-in.

FUNCTION

This is used to associate a particular function list with an access code. This can be used to limit the capabilities of the user of an access code. The < function-list-id > must match one of those declared in the FUNCTION section or "FMCP01". FMCP01 is the function-id associated with the access code supplied on the SECURITY card in the Cold Start deck. An access code with no FUNCTION clause can do only level-0 SPO commands (system status interrogation).

PRIV or PRIVILEGED

This makes the usercode a privileged user. A PRIVILEGED user also has security maintenance capabilities. This means that USERHO or USERLS can be run under the specified access code. The access code supplied on the SECURITY card in the Cold Start deck has this capability by default. More than one access code may be privileged. A privileged access code cannot be REMOVEd. However, the PRIV attribute can be CHANGEd.

When a common <media-list-id> or <function-list-id> is used for many access code specifications, these common ids can be declared in a STANDARD declaration. The format of a STANDARD declaration is:

```
STANDARD MEDIA <media-list-id>;
STANDARD FUNCTION <function-list-id>;
```

A STANDARD declaration is effective for all subsequent access code specifications which do not declare MEDIA or FUNCTION until end-of-input or another STANDARD declaration. When an access code specification declares MEDIA or FUNCTION, the STANDARD parameter is not applied.

Example 1:

```
MEDIA SECTION;
 ADD MINE PROCESSOR 0 1 HARDWARE OCS, HARDWARE CRD,
    STATION TERM06;
 ADD YOURS PROCESSOR 2 HARDWARE OCS, UNITID REMTWX, UNIT 24/0;
 CHANGE OTHERS HARDWARE OCS;

FUNCTION SECTION;
MYCANDO SECURITYCLASS PRIVATE, LEVEL 8, APP 3,
    CANDECONTROLLER, LIBMAINT, CTLD;
ADD MYOCS LEVEL 9;
YOUCANDO SECURITYCLASS PUBLIC, LEVEL 5, NOT LIBMAINT, CTLD;

YOUROCS LEVEL 9;

ACCESSCODES SECTION;
ADD 8485; PW ITSME NO CHANGE; CG N/R; PRIVILEGED;
    UN JOHNSMITH; MEDIA MINE; FUNCTION MYCANDO;
ADD UC ROOM407; PW SECRETARY; DEFAULT 567;
    MEDIA MINE; FUNCTION MYCANDO
```

In example 1, the media lists MINE and YOURS are associated with their respective hardware devices. The attributes previously associated with the media list called OTHERS are replaced with the single hardware type of OCS.

Access codes associated with the MYCANDO function will create private disk files unless otherwise specified. Keyboard commands entered will be assigned level 8. A CANDE user logged-in with this function-id will have default application mode 3; the user will also be able to create pseudo-card decks from the terminal as well as to load and dump files. The user will be identified to CANDE as a privileged user.

MYOCS is assigned SPO level 9, nothing else. Load and dump capabilities are not present because LIBMAINT is not specified.

Files created under the function-id YOUCANDO will be assigned security according to information in a guard file, in programmatic file attributes, or in file equate statements, otherwise, they will be PUBLIC.

The ACCESSCODES section declared only access codes for the media-id MINE and the function-id MYCANDO. The other media and functions are not mentioned and would have to be added in a later run of USERHO.

The following example shows how charge number validation could be implemented.

Example 2:

```
MEDIA SECTION;
FUNCTION SECTION;
 EVERYBODY SECURITYTYPE PUBLIC, LEVEL 5;
ACCESSCODES SECTION;
STANDARD FUNCTION EVERYBODY;
UC N/R; PW N/R; CG 345633;
UC N/R; PW N/R; CG 7648;
UC N/R; PW N/R; CG 87642;
```

In Example 2, only three valid charge numbers exist. Users can log-in with any usercode and password but the charge number must be one of the three. Programs could be executed as

```
? USER //<charge no.>; EXECUTE <program>
```

The slashes act as place holders to indicate that usercode and password are omitted.

USERLS

The USERLS program is used to list the contents of the USERFL file. USERLS can be executed using a PRIVILEGED access code or the access code specified on the MCP Cold Start SECURITY card. The output is a formatted description of the records currently in USERFL.

The following statement is used to run USERLS.

```
? USER <usercode>/<password>/<cg no.> EX USERLS
```

SECTION 4 OPERATING PROCEDURES

MCP REQUIREMENTS

To invoke disk file security, a SECURITY card must be included in the Cold Start deck. The Security option DFSC can only be set or reset during Cold Start. The format of the SECURITY card is

SECURITY <usercode>/<password>/<cg no.>

The access code declared on the SECURITY card is a privileged access code attached to the FMCP01 function-id. In brief, the FMCP01 function-id allows all media and contains all possible capabilities listed in the USERHO FUNCTION section. After Cold Start, USERHO can be executed to set up the USERFL only under this access code.

Any access code can be assigned to the FMCP01 function-id through USERHO.

The Security module of the MCP is loaded into memory when DFSC=1. The memory required is 14KD. This module must be present to handle the Security requests initiated by file manipulation and operator inputs.

The Disk File Security Directory (DFSD) is created during Cold Start when Security is invoked. Additional information is stored here for each file in the directory. The DFSD is located between the disk directory header blocks and the disk file headers.

For diskpack files, no additional structures are created since the security information is stored in the diskpack file headers.

The security information consists of the SECURITYTYPE, SECURITYUSE, SENSITIVEDATA file attributes, the usercode of the file creator, and if applicable, SECURITYGUARD, as described in the following paragraphs.

USERCODE
The usercode is a 1-to 10-character combination of upper/lower case letters and numbers used to identify the creator of the file or program.

SECURITYTYPE
SECURITYTYPE specifies who may access a file. Valid types are PRIVATE, PUBLIC, or GUARDED. For files created under a usercode, the default value is PRIVATE.

PRIVATE	Only the creator of the file may access the file in the manner specified by SECURITYUSE. The creator of a file, in this case, is a user with a matching access code.
PUBLIC	All users may access the file in the manner specified by SECURITYUSE.
GUARDED	A guard file, identified by SECURITYGUARD, must be accessed before a user may be given permission to use the file. The guard file specifies how the file can be used and includes a default SECURITYUSE for usercodes not listed in the guard file. If SECURITYGUARD is missing or if the guard file is not present, PRIVATE is invoked instead of GUARDED.

SECURITYUSE
SECURITYUSE specifies the manner in which a PRIVATE or PUBLIC file may be accessed. If SECURITYTYPE is GUARDED, the attribute is taken from the guard file. For files created under a usercode, the default is IO.

IN	Read-only access is allowed on the file.
IO	Read/Write access is allowed on the file.
OUT	Write-only access is allowed on the file.
SECURED	This file is a program and can only be executed if it passes the MCP requirements for a program. It cannot be opened as a data file by a user program.

SECURITYGUARD
This specifies the file-id of the guard file, if SECURITYTYPE is GUARDED.

SECURITYFAMILY
This specifies the multifile-id of the guard file when it is on 180-byte media. A multifile-id of DISK means the guard file is on 100-byte media.

SENSITIVEDATA
If this attribute is specified, the MCP intrinsic CLRSNS automatically overwrites all records in the file with a random data pattern.

FILE HANDLING

The following paragraphs discuss some aspects of file handling with Security.

Creating Secured Files

Security attributes may be applied to files in the following ways:

1. After Cold Start, by the use of ALLOCATE cards to apply security attributes to permanent disk files. Refer to Control Cards, later in this section.
2. By using File Equate cards with security attributes included on them. This method can be used to add security attributes to files created by programs compiled without Security or to change program defined security attributes. Attributes supplied by File Equate override those compiled into the program. Refer to Control Cards, later in this section.
3. By CANDE commands MAKE and ACCESS initiated by the creator of the file to add or change the security attributes for a permanent disk or diskpack file.
4. By running under a usercode where the default SECURITYTYPE is taken from the USERFL file and applied to all files and programs created under that usercode when security attributes are not specified.
5. By programs compiled by any compiler that places the security attributes into a Security Attribute Storage Area (SASA) and stores a pointer to the SASA in the FIB (FIBEXT).
6. By specifying the attributes on the compile card. Refer to Control Cards later in this section.
7. By using LOADMP, PACKUP or SYSTEM/COPY to load files from tape that have security attributes included with the files on tape.

In all but the last of the above cases, the usercode attached to the file is the usercode under which the creator is running. Once a file has been created, its usercode cannot be changed.

For LOADMP,PACKUP and SYSTEM/COPY, the files retain the creator's usercode and security attributes which have been dumped to the tape. The files do not assume the usercode from LOADMP, PACKUP, or SYSTEM/COPY.

Only the COBOL-74 compiler creates a SASA.

Security attributes can be attached to a file when it is created. A utility program called CHGSEC can be used for files which previously had no security attributes, from a system which did not have Security set. Refer to section 6 for details on CHGSEC.

Open Output Files

When Security is set, all files created are secured; that is, all files will be either PUBLIC, PRIVATE, or GUARDED. Even though the program does not specify explicit security attributes, these attributes are implied by the usercode under which the program is running. If security is not desired for a particular file, the usercode for the program should be one which has a default of SECURITYTYPE PUBLIC. The following sequence of events occurs whenever a new file is opened.

When the file is opened, the default SECURITYTYPE is taken from the USERFL file, and SECURITYUSE is set to IO.

The FIB is interrogated for a SASA pointer. If one is present, the MCP updates the SASA with any security attributes that may have been entered through a file equate card. The updated SASA attributes override all previous security attributes. If a SASA does not exist, the security attributes, if any, from the file equate card are used.

If neither a SASA nor a file equate card exists, the default values from the USERFL record apply.

The attributes are validated. For a diskpack file, the pack file header is updated with the security attributes. For a disk file, the security bit is set in the disk directory header entry for the file and an entry is made in the DFSD.

When the open is complete, the SASA can be programmatically examined to determine what final security attributes were applied to the file.

Open Permanent Files

For disk files, security attributes are stored in the DFSD. A disk file is secured if the security bit is set in the disk directory header. Disk pack file security attributes are stored in the diskpack file header. A diskpack file is secured if the SECURITYTYPE field is not empty.

Depending upon the value of SECURITYTYPE, one of the following actions will be taken:

PRIVATE
: If the usercode of the program attempting the open is the same as that attached to the file, access is allowed according to SECURITYUSE. If the usercodes are not equal the program is aborted.

GUARDED
: For a disk file, the guard-file-id is taken from the DFSD entry. For a diskpack file, the guard-file-id and the SECURITYFAMILY attributes are taken from the file header. A search is made for the guard file. If the specified guard file cannot be found, SECURITYTYPE PRIVATE is used.

The guard file is read and then searched for an entry applicable to the program performing the open. When found, the SECURITYUSE attribute is taken from the guard file entry.

If the guard file does not contain an applicable entry for the program, the default SECURITYUSE attribute in the guard file is used. If no default is specified, no access is granted.

PUBLIC
The SECURITYUSE attribute associated with the file is used.

The SECURITYUSE value is compared with the type of open being attempted by the program. If they are not compatible, the program is aborted.

For open input or I/O, if FIB-DA:4 is set, the security attributes of the file are copied into the SASA of the file, if there is one.

File Close

To be able to close a file, the program must have been able to open the file. This implies that the usercode for the program was acceptable. Therefore, the user of the file may close the file in any manner with the following exception. If close purge is requested and the file is PRIVATE, the user must be the creator of the file or a privileged user. If the file is PUBLIC and·IO, the file is purged. If the file is GUARDED, the use must be IO for PURGE to be allowed. File Close

When PURGE is allowed, and if the SENSITIVEDATA flag is set, the bound intrinsic CLRSNS is scheduled for execution under an MCP privileged usercode to overwrite the file with a random data pattern before the file is removed from the directory.

If an output file is to become a permanent file (CLOSE LOCK or RELEASE) a DUP LIB condition may occur. If the DUP LIB condition results, then the creator of the new file has a usercode which is privileged, has LIBMAINT capability, or matches the usercode of the old file.

If the usercode for the new file meets none of these criteria then instead of a DUP LIB condition, the offending program is aborted with a non-matching usercode error.

If the RMOV MCP option is set under the above DUP LIB conditions, the DUP LIB is automatically removed or the error occurs.

LIBRARY MAINTENANCE

The intrinsic versions of LOADMP, PACKUP, and SYSTEM/COPY are capable of handling files that have security attributes. The security attributes are dumped to tape with the files and are reestablished when the files are reloaded. The previously mentioned interface areas, SASA, FIBEXT, FIB-DA:4, are used.

Pre-file security versions of LOADMP and PACKUP are capable of loading all files dumped from a system using security because the SASA, FIBEXT, and FIB-DA:4 are not used in these versions of the utilities. However, PRIVATE and GUARDED files dumped to tape cannot be reloaded to a non-secured system running MCP 6.6 or greater. PUBLIC files are not affected by this restriction.

CONTROL CARDS

The remainder of this section describes the control cards used with Security.

File Equate Card

The File Equate feature can be used at run time to change certain file attributes. The security options follow all other file equate parameters (refer to the MCP System Software Operations Guide, vol. 1). The security option syntax is shown in figure 4-1.

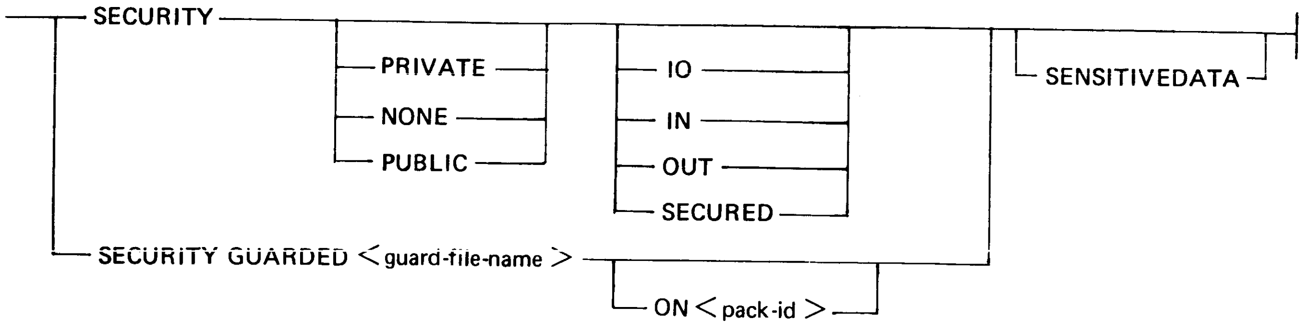


Figure 4-1. Security Options – File Equate Card

For example:

```
FILE OTFILE NEEWID DPK SECURITY PRIVATE IO
FILE FILEX MYOWN SECURITY GUARDED GFILE SENSITIVEDATA
```

Compile Card

To apply security attributes to the code file generated by a compiler, security options may be included in the COMPILE statement. The security options may follow any other options on the compile card (refer to the MCP Software Operational Guide, vol. 1). A semicolon must delimit the security attributes from any succeeding attributes. The security options are the same as for the File Equate card.

SECURED is the only valid attribute to secure the code file so that it can only be executed and not updated as a data file. If an option other than SECURED is requested, the code file can be treated as a data file.

For example:

```
CMP PROG COBOLV LIB SECURITY PUBLIC SECURED
CMP XXX COBOL LIB SECURITY PRIVATE IO
```

ALLOCATE Card

Security attributes can be assigned to ALLOCATEd files, but this can be done only on a post-Cold Start ALLOCATE. The security options follow all other options in the ALLOCATE statement. The syntax is given in figure 4-2.

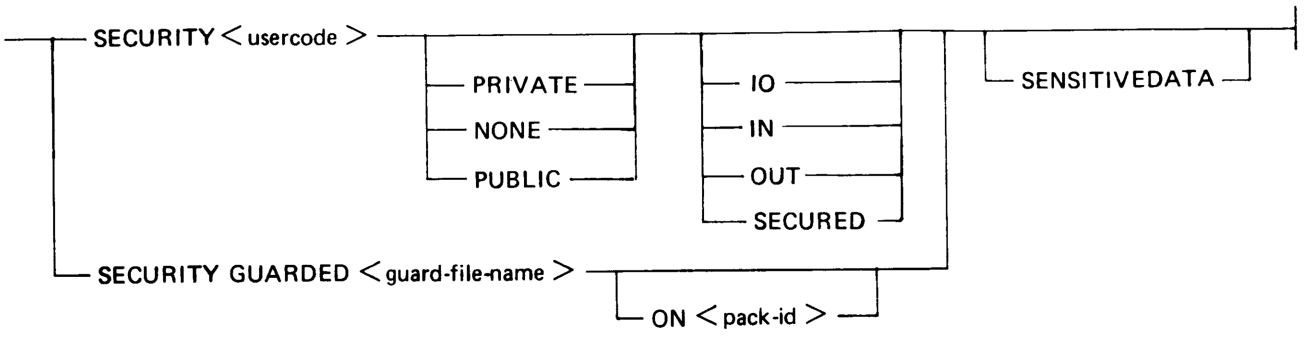


Figure 4-2. Security Options – ALLOCATE Statement

CHRG OPTION

Many installations have the MCP option CHRG set to require charge numbers with program executions. Security can be used to validate the charge numbers that are entered. The setting of each option is independent; that is, either can be set without the other. Setting Security can eliminate the need for setting the CHRG option. Setting both options can create situations such as the following.

1. If the usercode specifies charge number N/R (not required), then the user can log-in with or without a charge number. The log-in charge number, if supplied, is used for all job executions initiated by this usercode. Additionally, a different charge number may be supplied with the execute statement, overriding the charge number from the log-in. This overriding charge number is the one entered in the RLOG. Jobs are not scheduled unless a charge number is supplied. This method does not validate charge numbers.
2. If the usercode specifies charge number N/A (not allowed), then the log-in must not contain a charge number. Furthermore, job executions which supply a charge number on the EXECUTE statement are not allowed (due to N/A). This usercode cannot execute any jobs if CHRG is set. It may perform other functions but it cannot execute programs.
3. If the usercode allows a masked charge number, then the charge number supplied with the log-in will be attached to all program executions. A charge number may be supplied with the program initiation but, to be valid, that charge number must conform to the masking specifications from USERHO.
4. If the usercode requires a specific charge number, then only that charge number can be used on program executions. Any attempt to change from the log-in charge number will be rejected. This method allows complete validation of charge numbers.

In brief, Security validates charge numbers which are entered through log-in or job execution. The charge numbers must meet the specifications defined to USERHO. Failure to meet these parameters results in rejection of the input request.

The CHRG option need not be set for charge number requirements on programs. Since a charge number is attached to all jobs initiated under a usercode, specification of the valid charge numbers to USERHO ensures that only valid charge numbers are logged-in the Run Log.

SECTION 5
UTILITIES

Certain system utilities have application with Security. These utilities, such as GUARD, CHGSEC, and CLRSNS, are discussed in the this section.

GUARD

The system intrinsic GUARD is used to create the optional guard file which provides defined levels of privacy to permanent disk and diskpack files for various users. These levels specify which usercodes and programs can have access to files as well as the type of access allowed.

There may be more than one guard file on the system. A guard file is not assigned to a particular file and may be common to more than one file. The guard file is referenced when SECURITYTYPE is GUARDED. SECURITYGUARD indicates the id of the guard file.

The input to GUARD describes the access rights for one or more users or for one or more programs. If multiple input requests exist referring to the same usercode or program, the first input request is used.

The input syntax to GUARD is illustrated in figures 5-1 to 5-6.

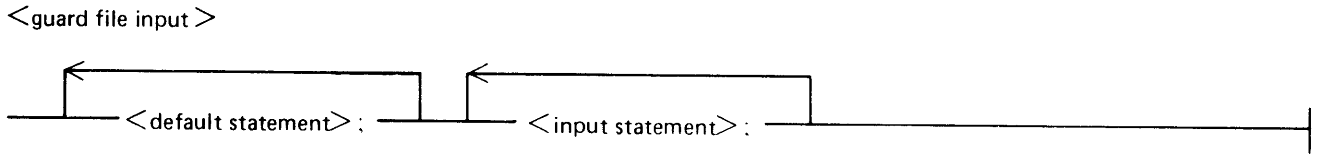


Figure 5-1. GUARD File Input Syntax

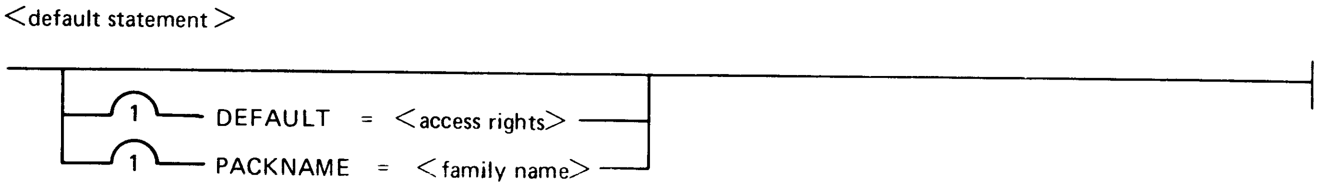


Figure 5-2. <default statement> Syntax

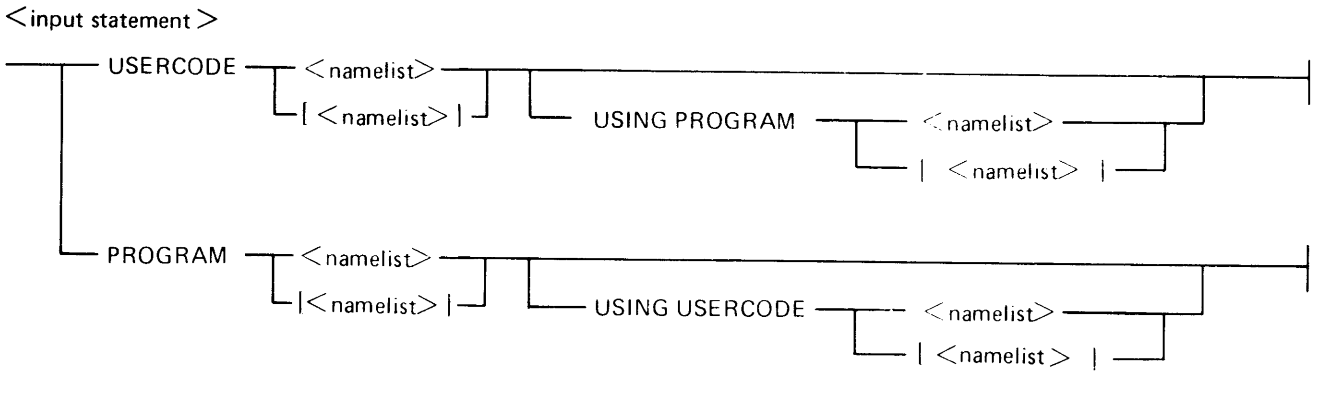


Figure 5-3. <input statement> Syntax

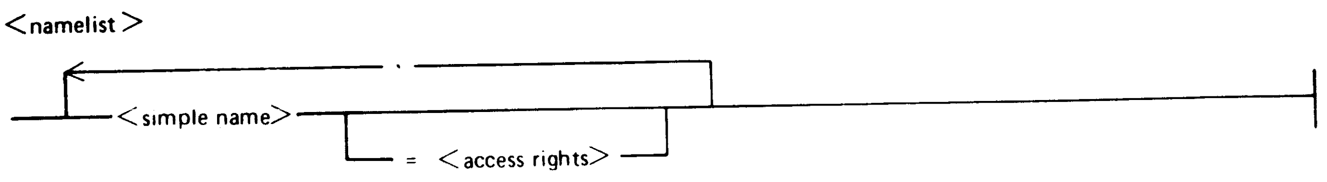


Figure 5-4. <namelist> Syntax

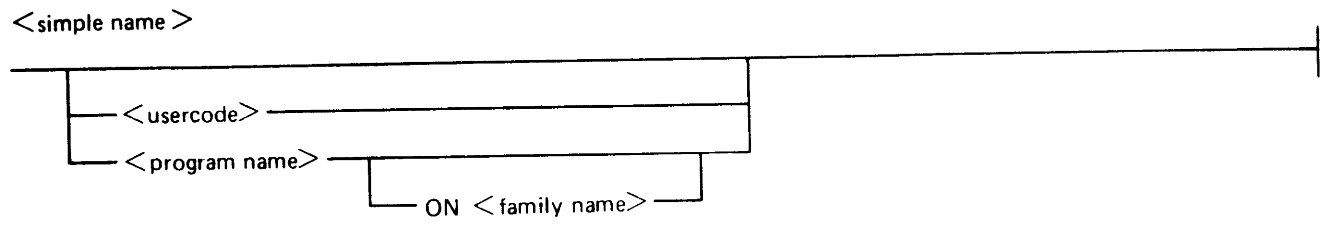


Figure 5-5. <simple name> Syntax

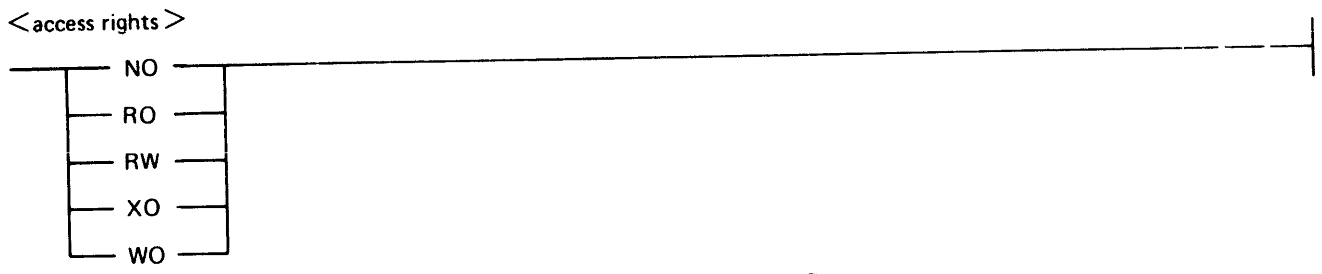


Figure 5-6. <access rights> Syntax

Semantics

The DEFAULT access right applies to any program or usercode not named in the guard file. Individual entries for which no access rights are given also use the DEFAULT value.

PACKNAME specifies the family name to be used for every program without an ON <family name> clause.

If no default statement is used, the access right assumed is NO (no access) and <family name> is DISK (100-byte media).

The USING clause enables the user to specify combinations of programs and usercodes. Certain programs may be coded such that one user may use them for reading files (input only) under a certain usercode, while another user (under a different usercode) is allowed to update files with the same programs. In the event of conflicting input requests, the first one given is used.

Access rights are defined as follows:

NO	No access is allowed.
RO	Read-only access is allowed.
RW	Read-write access is allowed.
XO	Execute-only access is allowed.
WO	Write only access is allowed.

The <family name> should be the name of a diskpack or DISK. The diskpack name refers to a diskpack from which the specified program is executed. DISK is the same as no family name specified.

If the program entry has a pack name other than DISK, then any program which initiates the guard file search must reside on the specified pack. Thus, if the guard file specified B ON PCK, but B is executed from OTHPCK, no access is granted. However, if the program entry does not have a pack name, or if the pack name is DISK, then the program which initiates the guard file search may reside on any media.

In the following examples, each statement is immediately followed by a brief description of its function.

Examples:

USERCODE AXER = RO;
User AXER may only read the file.

USERCODE SSST USING PROGRAM SUMIT = RW;
User SSST may read or write the file but only using program SUMIT.

PROGRAM JLZ = RO USING USERCODE BLT = RW;
Users may only access the file through program JLZ.

Only BLT may use JLZ to read or write the file. Other users are restricted to read-only.

USERCODE [AA, BB, CC = XO];
AA, BB, and CC all have execute only access rights.

USERCODE [AA = RO, BB, CC = RW];
AA has read-only rights. BB and CC have read-write access.

• USERCODE [U, V] USING PROGRAM [X, Y = RO, Z = RW];
Both U and V have the same access rights. They may have read-only access using program X and Y, or read-write access using program Z. Otherwise, they have the default access right.

PROGRAM [B, C = RO, D, E, F = RW];
Programs B and C are assigned an access right of read-only. Programs D, E, and F have read-write access rights.

PROGRAM [B,C, D = RO] USING USERCODE [USRA, USRB = RW];
Programs B, C, and D may access the file on a read-only basis unless they are executed with usercodes USRA or USRB, in which case they may access the file on a read-write basis.

PROGRAM A ON PCK = RO;
Program A executed from pack PCK has read-only access rights.

MCP Handling

When a file is secured by a guard file, every open of that file causes the MCP to examine the access rules in the guard file before granting or denying access.

The guard file may be created before or after it is attached to a file. If the guard file is not found at open time, the file is treated as if SECURITYTYPE were PRIVATE.

Operating Instructions

Input to the GUARD program consists of cards in a file called CARD. A print file is produced which shows syntax errors as well as the input statements.

The output of GUARD is a guard file. The file will be written on diskpack unless label equate is used to force the hardware type to DSK. The default guard-file-id is GARDFL. For label equate purposes the internal-id of the guard file is GUARD.

Example:

```
? USER <UC>/<PW>/<CG>
? EX GUARD
? FILE GUARD = MYGARD DSK
? DATA CARD
<input cards>
? END
```

GUARD can also be used to list an existing guard file as follows:

```
? USER <UC>/<PW>/<CG>
? EX GUARD/L
? FILE GUARD = MYPACK/GARDME. (Guard file on pack)
```

No data cards are required to list a guard file.

Usercodes may be one to ten characters in length. File and family names may be from one to six characters in length. Identifiers longer than allowed will not be syntaxed but will be truncated to the indicated maximums. A guard file is not created if syntax errors are encountered.

Table 5-1 contains the layout of a guard file record.

Table 5-1. Guard File Record Layout

Location, Size (digits)	**Description**
0,12	Guard-file-type identifier (A hash total of the numeric of the first and second halves of the usercode, and the default family name).
12,20	Usercode of the file creator
32,12	Default family name
44,1	Default access right
45,27	Filler

Table 5-1. Guard File Record Layout (continued)

Location, Size (digits)	Description
72,72	First entry 10 UA Usercode 6 UA Program-id 6 UA Securityfamily 6 UA Reserved 6 UA Reserved 1 UN Access right 3 UN Filler
144,n	Additional entries (The final active entry is followed by an entry with a usercode of all hex F.)

CHGSEC

The CHGSEC utility changes the security attributes of a disk or pack file in order to convert non-secured files to secured files. Access to these files can then be controlled under Security.

Input to CHGSEC is from cards. A data card consists of three to six entries in free format describing the security attributes to be assigned to a file. Multiple files may be changed in one execution of CHGSEC. The usercode for each file is not specified as input, but rather is taken from the accesscode under which CHGSEC was executed. A print file is created listing all changes or syntax errors that occurred on a run. The syntax is given in figures 5-7 through 5-11.

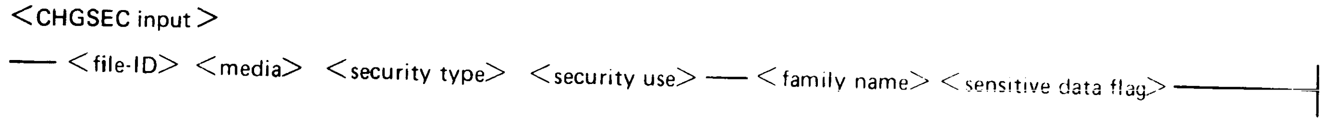


Figure 5-7. CHGSEC Input Syntax

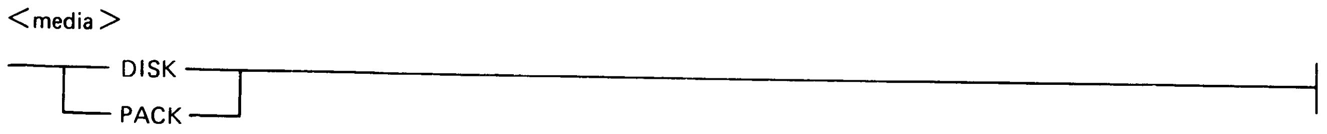


Figure 5-8. <media> syntax

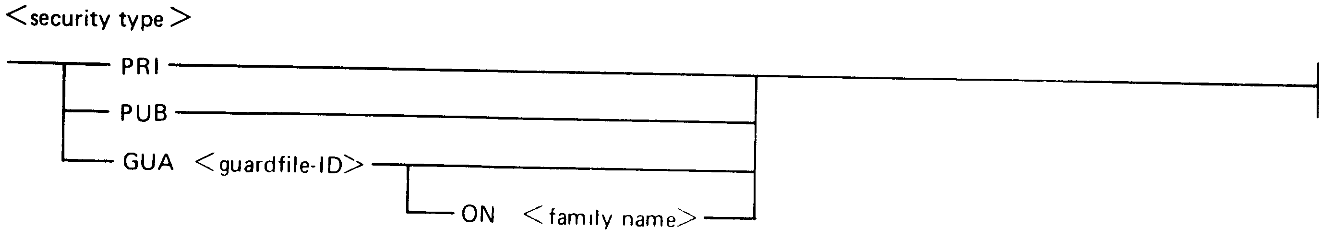


Figure 5-9. <security type> syntax

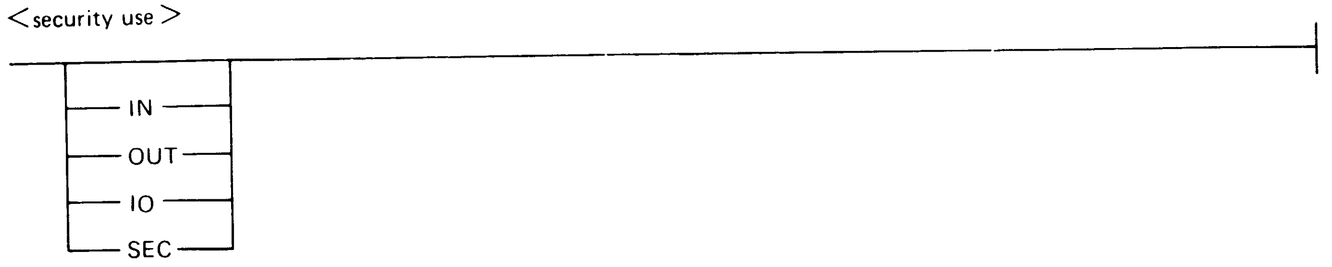


Figure 5-10. <security use> syntax

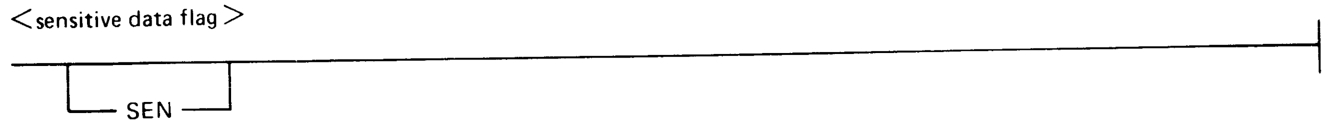


Figure 5-11. <sensitive data flag> Syntax

Semantics

<file-id> specifies the name of the file whose attributes are to be changed.

<media> specifies the location of the file whether on disk or on diskpack.

<security type> is the new SECURITYTYPE to be assigned to the file. For disk files which are GUARDED the guard file must be on disk (100-byte media).

<security use> is the new SECURITYUSE to be assigned to the file. This is omitted if SECURITYTYPE is GUARDED.

<guardfile-id> is the name of the new guard file.

The <family name> clause is used to designate the pack on which the file resides. The ON <family name> clause designates the pack for the guard file.

The <sensitive data flag>, when used, specifies that before removing the file from the disk or disk-pack directory, the file areas are to be overwritten by a random data pattern. This prevents the data from being used after the file has been removed.

An example card deck for CHGSEC follows:

```
? USER <UC>/<PW>/CG>
? EX CHGSEC
? FILE PRINT = FUL96/CHGSEC
? DATA CARDS
DSKFIL DISK GUA GRDFIL
DPKFIL PACK GUA GRDDPK ON SYSTEM
@00185 PACK PRI IO
MYFILE PACK GUA GRDPK2 ON MYPACK SEN
? END
```

The data files must be present when CHGSEC is run. The Disk File Security (DFSC) option must also be set.

CLRSNS

The CLRSNS intrinsic is used to destroy the data in files assigned SENSITIVEDATA when these files are removed. CLRSNS is automatically scheduled for execution whenever a SENSITIVEDATA file is removed. The data in the file is overwritten with a random data pattern, then the file areas are returned to available space. CLRSNS is executed under an MCP privileged usercode which allows it to access any file.

No user interaction is required to initiate CLRSNS other other than the specification of SENSITIVE-DATA.

DIRECT I/O UTILITIES

Utility programs such as DISPKV and DSKOUT which use Direct I/O in their normal execution do not function unless they arc executed under a usercode which has DIRECTIO associated with it.

Field Engineering maintenance/test routines also fall into this category.

SECTION 6
LANGUAGE REQUIREMENTS

Certain compilers have requirements affecting Security. These are COBOL ANSI-74, FORTRAN ANSI-77, CANDE, and Pascal. This section describes the special requirements of the various languages.

COBOL ANSI-74

COBOL-74 programs use the VALUE OF clause to specify security attributes. If at least one of the following VALUE clauses is present in a source program, a SASA is created by the compiler to store the security attributes. The address of this field is stored in FIBEXT. If none of the VALUE clauses is declared, no SASA is created, and FIBEXT is set to zero.

The VALUE clauses are shown in figure 6-1.

```
                                [ PRIVATE  ]
VALUE OF SECURITYTYPE IS        [ GUARDED  ]
                                [ PUBLIC   ]

VALUE OF SECURITYUSE IS         [ SECURED  ]
                                [ IN       ]
                                [ OUT      ]
                                [ IO       ]

VALUE OF SECURITYGUARD IS          [ Guard file-id ]

VALUE OF SECURITYFAMILY IS         [ Family name   ]

                                   [ TRUE   ]
VALUE OF SENSITIVEDATA IS          [ FALSE  ]
```

Figure 6-1. COBOL-74 VALUE Clauses

If some of the VALUE clauses are not specified, default values are applied for the attributes which are not declared. The default attributes are given in Table 6-1.

Table 6-1. COBOL-74 Security Attribute Defaults

Attribute	Value
SECURITYTYPE	PRIVATE
SECURITYUSE	IO
SECURITYGUARD	" "
SECURITYFAMILY	DISK
SENSITIVEDATA	FALSE

The security attributes of a file can be changed by the program before the file is opened. This is accomplished with the COBOL statement CHANGE ATTRIBUTE (refer to the COBOL ANSI -74 Language Manual, form no. 1090735).

Example 1:

```
CHANGE ATTRIBUTE SECURITYUSE OF FILEXX TO IN.
```

After the file is opened, the SASA may be examined to ascertain the final security attributes which have been applied to the file.

Example 2:

```
IF SENSITIVEDATA IS FALSE .....
IF SECURITYTYPE IS PUBLIC .....
```

CANDE

The SECURITY or ACCESS command allows a user to change the security attributes of a file. Only SECURITYTYPE and SECURITYUSE can be specified. The syntax is given in figure 6-2.

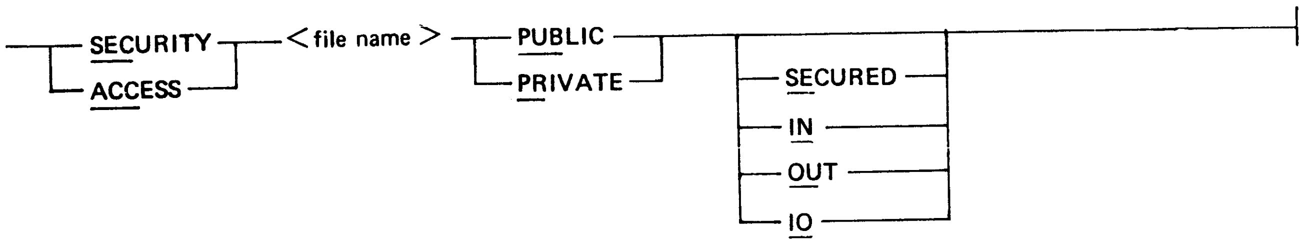


Figure 6-2. SECURITY/ACCESS Command Syntax

If neither SECURITY nor ACCESS is specified, the default is taken from the USERHO specification for the usercode.

FORTRAN ANSI-77

FORTRAN-77 programs use the FILE and OPEN statements to specify security attributes. Valid attributes with corresponding values are given in table 6-2.

Table 6-2. Security Attributes with FORTRAN-77

Attribute	Value
SECURITYTYPE	DEFAULT
	GUARDED
	NONE
	PRIVATE
	PUBLIC
SECURITYUSE	IN
	IO
	OUT
	SECURED
SECURITYGUARD	<string>

Table 6-2. Security Attributes with FORTRAN-77
(continued)

Attribute	Value
SECURITYFAMILY	<string>
SENSITIVEDATA	TRUE FALSE

Following are examples of the use of security attributes with FILE and OPEN in FORTRAN-77.

```
FILE 4(KIND = "DISK", SECURITYTYPE = "PRIVATE")

OPEN (4, SECURITYTYPE = "GUARDED", SECURITYGUARD = "GRDFIL", SECURITY-
FAMILY = "DISK")
```

PASCAL

PASCAL programs specify security attributes with the Attribute Specification clause. The file attributes and defaults are shown in table 6-3.

Table 6-3. Security Attribute Values with PASCAL

Attribute	Value	Defaults
SECURITYTYPE	GUARDED PRIVATE PUBLIC	DEFAULT
SECURITYUSE	IN IO OUT	IO
SECURITYGUARD	6-character string	'DISK'
SECURITYFAMILY	6-character string	blanks
SENSITIVEDATA	TRUE FALSE	FALSE

An example of PASCAL syntax follows.

```
PROGRAM Example
   (DPKFIL : FILE <KIND = DISKPACK
          SECURITYTYPE = PUBLIC>);
```

Security attributes can also be accessed and specified in PASCAL with the procedures GETFILEAT-TRIBUTE and SETFILEATTRIBUTE. The SECURITYTYPE attribute cannot be set to default.

INDEX

INDEX (Cont.)

Documentation Evaluation Form

Title: B 2000/B 3000/B 4000 System Security Installation and Operation Guide

Form No: 1127362

Date: November 1982

Burroughs Corporation is interested in receiving your comments and suggestions, regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

____

____

____

____

____

____

____

____

____

____

____

____

____

____

From:

Name ____

Title ____

Company ____

Address ____

____

Phone Number ____________________ Date ______________

Remove form and mail to:

Burroughs Corporation
Documentation Dept., TIO - West
1300 John Reed Court
City of Industry, CA 91745
U.S.A.

Printed in U.S. America November 1982 1127362