GETTING STARTED WITH BiiN™

BiiN™

BiiN™

GETTING STARTED WITH BiiN™

Order Code: 6AN9000-1AJ00-0BA2

LIMITED DISTRIBUTION MANUAL

This manual is for customers who receive preliminary versions of this product. It may contain material subject to change.

BiiN™
2111 NE 25th Ave.
Hillsboro, OR 97124

REV.	REVISION HISTORY	DATE
-001	Preliminary Edition	7/88

BiiN™ MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Additional copies of this or any other BiiN™ manuals are available from:

BiiN™ Corporate Literature Dept.
2111 NE 25th Ave.
Hillsboro, OR 97124

PREFACE

Purpose

This manual is a hands-on tutorial that shows how to enter the most common BiiN™ commands for daily work. A companion volume, the *BiiN™ Systems Commands Reference Manual*, contains complete descriptions of the commands used in this manual.

Audience

This manual is for all first-time users of the BiiN™ system who will be using the native interface, CLEX.

If you are using the BiiN™/UX interface to the system, refer to the manual *Introduction to BiiN™/UX* for beginning information.

Organization

This manual contains the following tutorial chapters and appendixes:

1. Welcome to BiiN™
How to logon, enter some commands, and get help.

2. Working with Files and Directories
How to show, copy, rename, and remove files and directories.

3. Your User Account
How to change your password, list your user profile, and customize CLEX.

4. Printing Files How to use the print queue.

5. Controlling Jobs
How to start and stop jobs, list previous commands, and redo previous commands.

6. Working with Windows
How to open a new window, work between windows, and close a window.

7. Protecting Files and Other Objects
How to control access to your files, directories, and other objects.

Appendix A, Command Quick Reference
Contains the name, synopsis, and syntax for the commands used in this manual.

Appendix B, BiiN™/UX Commands and BiiN™ Equivalents
Shows which BiiN™ commands are equivalent to Unix commands, to help UNIX-literate readers assimilate the system quickly.

Appendix C, Summary of Window Commands
Lists the commands that control windows on character terminals.

Appendix D, Roadmap to BiiN™ Documentation

Shows the BiiN™ document set with paths showing recommended reading sequence.

Notation

This manual uses the following notation:

`logoff`	Typewriter font shows command names, file names, and other system names.
`get.time`	Boxes surround your input (what you type).
window	Italic font shows a new term.
<Return>	Angle brackets surround keyboard keys. That is, if **<Return>** is shown, press the RETURN key on the keyboard.
<Ctrl-Z>	Angle brackets surround control keys. You must hold down the **<Ctrl>** key, press the **<z>** key, and then release both keys.

Related Publications

You may find the following manuals useful when learning about the BiiN™ system.

BiiN™ Systems Overview
: An overview of BiiN™ hardware and software benefits and features.

BiiN™ Systems Programmer's Guide
: General concepts and programming techniques for BiiN™ software development.

BiiN™ Command Language Executive Guide
: Tutorials on the BiiN™ command interpreter CLEX and command language BiiN™ CL.

BiiN™ Systems Commands Reference Manual
: Complete reference for BiiN™ CL commands.

CONTENTS

Chapter 1. Welcome to BiiN™

Chapter 2. Working with Files and Directories

Chapter 3. Your User Account

Chapter 4. Printing Files

Chapter 5. Controlling Jobs

Chapter 6. Working with Windows

Chapter 7. Protecting Files and Other Objects

Appendix A. Command Quick Reference

List of Figures

List of Tables

1 WELCOME TO BiiN™

Welcome to computing with BiiN™! This manual shows you how to enter the commands you will need for daily work on the system. Each of the chapters is a session that takes about 10 minutes to go through.

What You Need. These sessions assume you have the following prerequisites (see your system administrator if you need help):

- An installed BiiN™ system with a terminal (Fig. 1-1).
- A user account for yourself, including logon name and password. (If your account is not new, your displays may differ from some of the examples in this book.)
- A printer installed and ready (for the "Printing" chapter).

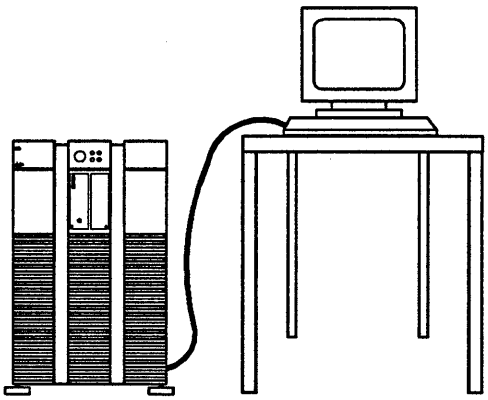


Figure 1-1. BiiN™ System and Terminal

This first session shows some basics about BiiN™ commands. After finishing this session, you will know how to:

- Logon and logoff
- Get the time
- List your home directory
- See who's on the system
- Abbreviate commands and get help

1.1 Logging On

To gain access to the system, you logon at the terminal. (If you have problems logging on, see your system administrator.) Press **<Return>** to get the logon prompt:

```
Press Return to continue.  <Return>
```

Enter your logon name and password at the prompts (use **<BACKSPACE>** to correct typing mistakes):

```
Logon name: joe <Return>
Password:   newuser <Return>
```

The system does *not* echo your password as you enter it. This makes it hard for you to see typing mistakes but it prevents others from reading your password.

Next, you see welcome messages from the system administrator. This part of the logon process may vary depending on what your system administrator sets up.

The following display shows a typical logon sequence.

```
   1  Logon Service


Please enter your identification.

Logon Name:    joe

Password:      newuser                    (not echoed)

... logon messages from system administrator...

Home directory:  ///org/dom/vs/users/joe

NoS          On since           User Name
   1  1988-05-09 14:04:48.17  ///org/dom/id/joe

Select one of the following terminal types,
  followed by a return:
     w: wyse 50/60
     v: vt102
     f: freedom 100
     ...
  :value=<derived>          -> w

Terminal type wyse 50.

Logon_CLEX - W: The terminal type has been changed
                within the logon script...

                Hit return to continue
```

Here, the administrator gives reminders about using the system, the home directory name, and status of this logon session. You may be asked to enter a terminal type, such as Wyse 50.

The duration between when you log on and when you log off is called a *session.*

1.2 Your Initial CLEX Window

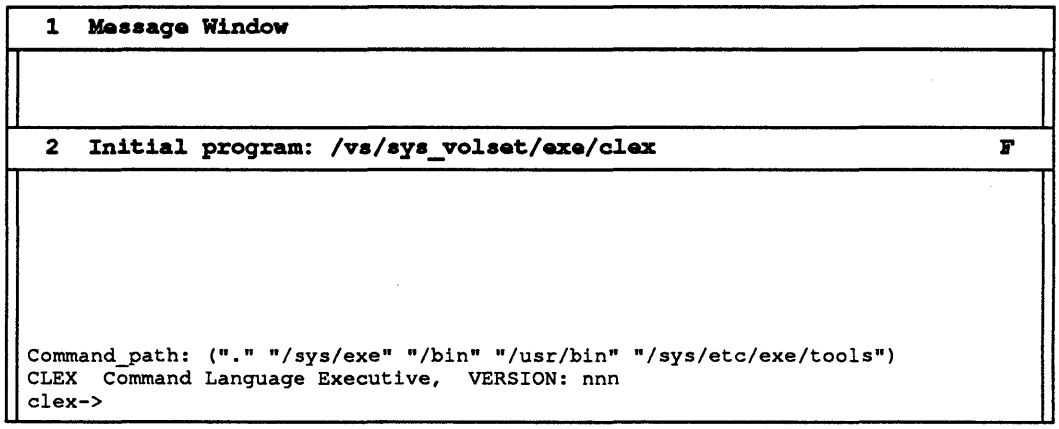
The initial screen after logon is divided into two *windows*, a message window and a CLEX window.

The smaller window, `Message Window`, is a read-only window that displays the system's messages to you. You do *not* type into the message window.

The larger window, `Initial Program`, runs CLEX when you logon. CLEX (Command Language Executive) is a *command interpreter*, a program that accepts and interprets your commands.

The CLEX window may contain some messages from the system administrator followed by the prompt `clex->`. (If your `Initial Program` window does not have the `clex->` prompt, see your system administrator. You will need to be set up with CLEX to use this manual.)

See the following display.



1.3 Logging Off

Whenever you want to end a logon session, enter `logoff`, then press **<Return>**. (Don't do it now if you want to continue.)

If you switch off the terminal without a `logoff`, your session is not ended although the screen is dark. You must always log off before switching off the terminal, or another person will be able to use your account without permission.

1.4 Recovering From Mistakes

You can use **<BACKSPACE>** and **<Ctrl-C>** to recover from mistakes when you type CLEX commands.

To erase the previous character (to the left of the cursor):

```
clex-> g<BACKSPACE>
```

To cancel the command in progress:

```
clex-> get.time <Ctrl-C>
clex->
```

You can use **<Ctrl-C>** or **<DEL>** any time to get back to the `clex->` prompt.

1.5 Getting the Time

To enter CLEX commands, enter the command, then **<Return>**.

For example, the `get.time` command displays the current date and time:

```
clex-> get.time
1988-04-01 16:06:42.78
```

1.6 Listing Contents of Your Home Directory

When you first logon, you are working in your *home directory*.

Use the command `list.object` to see the entries in your home directory:

```
clex-> list.object
.default_authority    .mail_AL      var_groups
.logon_script         startup
```

As a new user, your home directory contains two authority lists, two directories, and a script. The `.default_authority` authority list protects the objects in your account, and `.mail_AL` protects your mail. The `startup` directory contains a file of commands to be executed when you start various programs. The `var_groups` directory holds values for CLEX group variables, which can be used to customize CLEX commands. The `.logon_script` script contains commands that are executed when you logon. You will work with authority lists, startup files, and group variables later in these sessions.

Your home directory is the top of your directory hierarchy. You can store your personal files and other working data either in your home directory or in subdirectories that you create; thus, you can think of your home directory as your own personal real estate in the larger system.

The BiiN™ system uses directories to store names (entries) for objects on disk. An object is simply a BiiN™ container for data or programs: a file, a load image, another directory, and so on.

Directories are arranged in a hierarchy with some directories containing sub-directories. The top of the hierarchy is the *slash* (/) directory. *Pathnames* identify directory entries. (The name suggests a "path" from the top of the directory structure to the entry.) Slashes separate directory names. For example, if your home directory is `/users/joe`, directory "slash" (/) contains the directory `users` which contains the directory `joe`.

1.7 Abbreviating Commands

You can abbreviate command names for easier typing.

BiiN™ commands usually have two words separated by a period (as in `verb.noun`). A useful approach is to use the first three letters of each word. The only restriction is that you must enter the dot if the command name includes a dot.

For example, to list your home directory using abbreviation:

```
clex-> lis.obj
.default_authority      .mail_AL                startup
.logon_script           var_groups
```

If the abbreviation is too short and there is more than one possible command, you'll get a message that the command is ambiguous; but if you use the first three characters of each word, you will usually get a unique abbreviation.

For example, abbreviating `list.session_user` to `lis.ses` is ambiguous:

```
clex-> lis.ses
```

The system sends a syntax error message, and displays the choices:

```
lis.ses
^^^^^^^
  ***  SYNTAX ERROR:  ambiguous command name:

I list.session_system   I list.session_user
```

You can then try again with a different abbreviation.

In this case, you can use another abbreviation feature: names on each side of an underscore can be abbreviated. For example, a different, unique abbreviation for `list.session_user` is:

```
clex-> lis.s_u
user:                   ///org/dom/id/joe
session:                joe on tty017
status:                 active
------------------------------------------------------------
```

From here on in these sessions, abbreviations are shown after long command examples.

1.8 Seeing Who's On the System

You can use the `list.status` command to see who is logged on to the system:

```
clex-> list.status user :admin
///org/dom/id/normal_OS
///org/dom/id/joe
///org/dom/id/sue
///org/dom/id/system
```

This example shows entering *arguments* to a command. Arguments affect the operation of a command, and a command may have zero or more arguments. This command has two: `user` and `:admin`.

When entering arguments, be sure to use spaces between the command name and arguments.

1.9 Getting Syntax Help on a Command

You can use the question mark (?) or double question mark (??) to show command syntax. In general, ? shows syntax only, and ?? shows syntax plus description.

For example, to display the syntax of any command, such as `list.spool_rank`, enter a space and question mark after the command name:

```
clex-> list.spool_rank ?
lsr [:queue=<pointer>:=$spool.queue]
CONTINUE CMD:
```

Abbreviation: `lis.s_r ?`

At this point, you are prompted to continue the command by entering arguments, or you can enter **<Ctrl-C>** to cancel the command:

```
CONTINUE CMD: <Ctrl-C>
```

The help display shows you the command name, followed by information about arguments: name, type, and default value. This example shows syntax for the command's one argument (`:queue=`), of type `pointer` with default value equal to the value in the BiiN™ CL variable `spool.queue`. (You will work with BiiN™ CL variables in Chapter 3.) The brackets ([ ]) show that the argument is optional, not mandatory.

Note that if you do enter an argument, the following syntax characters should *not* be entered:
`[ ] < > :=`
Also, you do not enter the type *name* such as `pointer`. Instead, you'll be entering a pointer *value*.

In addition to getting syntax help with a command, you can get syntax plus a description of what the command does with the double question mark (??):

```
clex-> list.spool_rank ??
lsr [:queue=<pointer> := $spool.queue]
   -- Description:
   -- Lists all files in a spool queue in rank order.
   --
   -- Includes the following:
   -- * status
   ...
more?(<blank> | <lf> | d | q)q
CONTINUE CMD: <Ctrl-C>
```

If the description continues for more than one screen, you can enter **<SPACEBAR>** to see the next screen, or q to quit the description. As before, you are then prompted for an argument, or you can enter **<Ctrl-C>** to get back to CLEX.

There are other features of the question mark command that you may want to experiment with at your leisure; see Appendix A for more information.

1.10 Session 1 Summary

- Press **<Return>** to get the logon prompt, if it's not showing.
- **<Return>** erases the character to the left of the cursor.
- **<Ctrl-C>** cancels the command in progress.
- `get.time` gets the current time.
- `list.object` lists the entries in a directory.
- Abbreviation allows you to abbreviate a command name or argument name, using the shortest string that uniquely identifies the name.
- `list.status user` shows system status, such as who is logged on.
- The question marks `?` and `??` can show command syntax.

WORKING WITH FILES AND DIRECTORIES 2

This chapter shows how to work with files and directories. After you finish this session, you'll know how to:

- Create a file
- Name files and other objects
- Show file contents
- Copy a file
- Rename a file
- Remove a file
- Get a long listing of your home directory
- Create a new directory
- Change your current directory
- Use pattern-matching on directory entries

2.1 Creating a File

You can create a simple file by redirecting command output into a file. For example, you can list the directory that contains BiiN™ commands (`/exe`), and save that list in a file.

To save the list of commands in file `templ`, use `list.object` with the > option:

```
clex-> list.object /exe > templ
```

Abbreviation: `l.ob /exe > templ`

Notice that output is not to the screen, but to the file.

The BiiN™ CL option to redirect command output is >. (BiiN™ CL (Command Language) is the high-level language used to construct CLEX commands.)

A BiiN™ CL option, like an argument, affects the operation of a command. However, where an argument is defined for an individual command, an option can be applied to any command for which it makes sense; in this example, you can redirect the output of any command that has output. Options are usually entered at the end of the command.

To confirm that `templ` is there, use `list.object`:

```
clex-> list.object
.default_authority   .mail_AL      templ
.logon_script        startup       var_groups
```

Abbreviation: `l.ob`

You can also create a file using the BiiN™ system text editor, Emacs (see the *BiiN™ Systems Emacs User's Guide*).

2.2 Naming Files, Directories, and Other Objects

When creating names for your files and other objects, the following guidelines are recommended.

1. Use any number of letters, numbers, underscores, and dots:

   `A-Z` (uppercase letters)
   `a-z` (lowercase letters)
   `0-9` numbers
   `_` underscore
   `.` dot

   For example, these are effective names:

   ```
   my_file
   test.2Feb.300
   once_is_not_enough
   ```

   Names are limited to 256 characters. Uppercase is distinct from lowercase; for example `my_file` is different from `My_file`.

2. ***Don't*** use any other characters, and avoid starting a name with a number.

   If you use any other characters, the character may be interpreted by CLEX or another program to mean something other than a name. For example, if you wanted to call a file `either/or`, CLEX would think it was directory `either` and entry `or`, because CLEX expects a slash to separate a directory and an entry. If you start a name with a number, CLEX interprets the value as a numeric argument instead of a string argument.

2.3 Showing File Contents

The `pg` command displays file contents. (Note: `pg` is a command from the BiiN™/UX interface. It will be replaced by a BiiN™ CL command at a later software release.)

To show the contents of `templ`, use `pg`:

```
clex-> pg templ

...
clex                              manage.library
cobol                             manage.messages
compile.terminfo_entry            manage.os_monitor
:q
clex->
```

If the file is longer than one screen, you'll get a colon prompt (`:`). You can enter **<Return>** to see the next screen, or `q` to quit.

You can also use the BiiN™/UX command `cat`, which displays a file without pausing for a page at a time.

2.4 Copying a File

You can use the command `copy.object` to copy one file to another. (A file is one kind of BiiN™ object.)

For example, to copy `temp1` to `temp2`:

```
clex-> copy.object temp1 temp2
```

Abbreviation: `cop.ob temp1 temp2`

You can confirm the copy with `list.object`:

```
clex-> list.object
.default_authority      .mail_AL        temp1           var_groups
.logon_script           startup         temp2
```

Abbreviation: `l.ob`

2.5 Renaming a File

The command `rename.object` renames files and other objects.

To rename `temp2` to `temp3`:

```
clex-> rename.object temp2 temp3
```

Abbreviation: `ren.ob temp2 temp3`

2.6 Removing a File

The command `remove.object` removes the directory entry for an object.

To remove `temp3`:

```
clex-> remove.object temp3
```

Abbreviation: `rem.ob temp3`

2.7 Getting a Long Listing of Your Home Directory

The long listing of a directory shows more information about each entry in the directory. To get a long listing, use `list.object` with its `:long` argument:

```
clex-> list.object :long
.                    02-09 08:47 directory   joe    umc      208

.:
.default_authority   02-09 08:49 authority_l joe    umc      108
.logon_script        02-09 08:49 file        joe    umc     1471
.mail_AL             02-09 08:49 authority_l system umc      228
startup              02-09 08:49 directory   joe    umc      172
temp1                02-09 08:49 file        joe    umc     2117
var_groups           02-09 08:49 directory   joe    umc      172
```

Abbreviation: `l.ob :l`

The long display shows a line for each entry:

`.`	Name of entry (dot stands for current directory).
`02-09 08:47`	Month, day, and time the object was created.
`directory`	The *type* of the object, in this case *directory* for the current directory (dot), and *authority* for the authority list.
`joe`	The owner of the object (you).
`umc`	The *access rights* to the object. Access rights are *use*, *modify*, and *control*. In general, you have all rights to the objects you own.
`208`	The size of the object in bytes.

This example shows an important way of entering arguments: by name. Until now, you have mostly entered *values* for the command arguments: `temp1` and `temp2` for `copy.object`.

Each argument also has a *name*. You can enter arguments by name as well as by value. In this example, the name of the argument is `:long`. The colon (:) is important and must be included.

If you give an argument's name, such as `:long`, it can be entered in any position in the command line. If not given argument names, CLEX interprets arguments according to their *position* in the command line; this is what is happening with a command such as `'remove.object temp3'`.

In general, it is easier to enter *both* name and value when you don't know the argument's position (e.g., `:long`). It is easier to enter *just* the value when you already know the argument position (e.g., `remove.object temp3`).

Also, there's a shortcut for boolean (true/false) arguments such as `:long`. Instead of entering `:long=true`, you can simply enter `:long` to toggle the default (from false to true in most cases).

2.8 Creating a New Directory

The `manage.directory` command allows you to create a new directory. `manage.directory` is the first two-level command (*utility*) that you have used in these sessions. A two-level command has an *invocation* command, like `manage.directory`, that you enter from the `clex->` prompt. Once you have entered, the prompt changes, and you can enter any of that utility's *runtime* commands.

To create a new directory named `personal`, first enter the utility `manage.directory`:

```
clex-> manage.directory
manage.directory =>
```

Abbreviation: `man.dir`

The prompt changes so you know you're in the utility.

Then, use the runtime command `create` to create a new directory named `personal`:

```
manage.directory => create personal
```

To exit from `manage.directory`, enter the `exit` runtime command:

```
manage.directory => exit
```

To confirm that the new directory is there, use `list.object :long`:

```
clex-> list.object :long
...
personal              02-09 09:01 directory joe     umc        172
...
```

Abbreviation: `l.ob :l`

2.9 Changing Current Directory

The *current directory* is the one you are currently working in. When you first logon, you are working in your *home directory*, the top of your directory hierarchy. The command `set.current_directory` can change your current directory.

To see the name of your current (home) directory, use `list.current_directory`:

```
clex-> list.current_directory
///org/dom/vs/users/joe
```

Abbreviation: `l.cu`

The pathname with three leading slashes (`///`) is the system's *full pathname* for your home directory. This is just another name for the directory. You may find it easier, when you need to enter a pathname for your home directory, to use the short form `~`. For user `joe`, `~` is a short name for directory `/users/joe`.

To change your current directory to `personal`:

```
clex-> set.current_directory personal
```

Abbreviation: `s.cu personal`

You are now "in" the new directory.

It is useful to change your current directory to a different directory when you will be working with entries in that new directory; you can list entries, rename, copy, and so on by simply typing entry names instead of full pathnames.

To confirm that your current directory has been changed:

```
clex-> list.current_directory
///org/dom/vs/users/joe/personal
```

Abbreviation: `l.cu`

Any time you want to change back to your home directory, use `set.current_directory` with no arguments:

```
clex-> set.current_directory
```

Abbreviation: `s.cu`

2.10 Using Pattern-Matching

When entering a name as an argument, you can specify a *pattern*. Only names matching the pattern will be acted upon.

To list all files starting with `temp` and ending with a single character:

```
clex-> list.object temp?
temp1
```

Abbreviation: `l.ob temp?`

Other pattern operators available are:

?	Matches any single character.
*	Matches zero or more characters.
[*xyz*]	Matches any of the single characters within brackets where *x*, *y*, and `z` are single characters.
[*a-z*]	Where *a* and *z* are single characters, matches all ASCII characters between *a* and *z*, including *a* and *z*. The match always fails if *z* is greater than *a* in ASCII collating sequence.
\	Escape character. "Turns off" the special meanings of pattern operators. Must precede any of `?, *, [, ]` that are to be matched. For example, to match a real question mark in a name, you would enter `\?` (It is best to avoid pattern operators in names anyway.)

Note: In general, any time a command expects an argument that is a name of type *string*, you can include pattern operators. However, you cannot use pattern operators to match BiiN™ CL runtime commands. Also, a single or double question mark in place of a name is recognized as a help command, not a pattern operator.

2.11 Session 2 Summary

- > is a BiiN™ CL option to redirect output.
- Names for files and other objects can be any length, should not start with a number, and should include only letters, numbers, underscore, and dot.
- `cat` shows file contents.
- `copy.object` copies an object such as a file.
- `rename.object` renames a directory entry for an object such as a file.
- `remove.object` removes a directory entry for an object such as a file.
- `list.object :long` shows a long listing of directory entries.
- `manage.directory` can be used to create a directory.
- `list.current_directory` shows the pathname of your working directory.
- `set.current_directory` changes your working directory.
- Pattern operators can be used to select names that match a certain pattern.

YOUR USER ACCOUNT 3

Your logon name identifies your *user account*, so called because system resources can be assigned for each user. Your account stores your personal files and other working data—your user account is your personal real estate in the larger system. Also, you can customize CLEX to suit your preferences using startup files, BiiN™ CL variables, and command aliases.

After you finish this session, you'll know how to:

- Change your password
- List your user profile
- Customize your prompt string
- Examine your startup files
- Change your command path
- Examine BiiN™ CL variables
- Create an alias for a command

3.1 Changing Your Password

Use the command `change.password` to change your password.

You will be prompted for your old and new passwords:

```
clex-> change.password
Old password:          newuser
New password:          sesame
Retype new password:   sesame
```

Abbreviation: `ch.pas`

As with your logon, the passwords are not echoed on the screen. Both new passwords must match. If you make a mistake and they don't match, try again.

Passwords are an important part of system security. After you use `change.password`, no one, not even the system administrator, knows what your password is, so no one can log on under your name. See your system administrator for further password guidelines for your system. Some common guidelines are:

- Change your password at random intervals.
- Don't write your password down, and don't give it out.
- Random characters are better than names, birthdays, or other strings that an intruder could guess correctly.

3.2 Listing Your User Profile

A user account contains a *user profile.* Your user profile contains your logon name, home directory, initial program, and other things unique to your account.

To list your user profile, use `list.user_profile`:

```
clex-> list.user_profile
user:                joe
home directory:      /users/joe
initial program:     /vs/sys_volset/exe/clex
------------------------------------------------------------
```

Abbreviation: `l.u_p`

This short listing contains the following parts (your system administrator assigns values that make sense for you):

user	Your logon name, for example `joe`.
home directory	"Where you are" in the system when you first logon. For example, `joe`'s home directory is `/users/joe`.
initial program	The program that is automatically invoked when you logon, typically `/vs/sys_volset/exe/clex`.

3.3 Customizing Your Prompt String

You can change your prompt string by changing a BiiN™ CL variable. By default, the `clex` prompt is `clex->`, and is stored in the BiiN™ CL variable `cli.prompt`.

You can change the prompt temporarily with `set.variable`. To change your prompt string:

```
clex-> set.variable cli.prompt "yes, dear? "
yes, dear?
```

Abbreviation: `set.var cli.prompt "yes, dear? "`

The prompt immediately changes to your new one. You need to enclose the string value in quotes (") because it contains special characters (spaces and question mark).

Enter another command just to see the new prompt again:

```
yes, dear? get.time
1988
yes, dear?
```

To change the prompt back to `clex->`:

```
yes, dear? set.variable cli.prompt "clex-> "
clex->
```

Abbreviation: `set.var`

In general, BiiN™ CL variables affect the way CLEX operates. Later in these sessions you'll examine the BiiN™ CL variables that you can change, and find out how to make the changes permanent (to take effect each time you logon).

3.4 Examining Your Startup Files

Your user account initially contains two *startup files* in your `startup` directory. A startup file contains CLEX commands that are executed automatically when you start a program (such as the logon program or CLEX).

To see the entries in your startup directory `~/startup`, use `list.object`:

```
clex-> list.object startup
startup

startup:
.default_authority clex      logon
```

The `~/startup/logon` file contains commands that are executed when you first logon. The `~/startup/clex` file contains commands that are executed when you start your logon CLEX.

To see the contents of the `logon` startup file, use `cat`:

```
clex-> cat startup/logon
set.command_path (. /sys/exe /bin /usr/bin /sys/etc/exe/tools)
echo ""
echo "Command_path: " :omit_LF
list.command_path
```

These commands set and display your command path.

To see the contents of the `clex` startup file, use `cat`:

```
clex-> cat startup/clex
set.alias cd set.current_directory
set.alias ls "/bin/ls -C"

set.variable pglob.name clex
set.variable cli.prompt "clex-> "
```

These commands set useful aliases and BiiN™ CL variables, including the initial prompt for CLEX.

In addition to the files in the directory `~/startup`, the file `.logon_script` in your home directory contains commands that are executed when you first logon.

You can use any BiiN™ text editor to change these startup files.

3.5 Changing Your Command Path

A command path is a list of directories. When you enter a command, CLEX searches through each of the directories in the command path, in order, to find the command. You have a default command path for your account, which you can show with `list.command_path`:

```
clex-> list.command_path
("." "/sys/exe" "/bin" "/usr/bin" "/sys/etc/exe/tools")
```

Abbreviation: `l.com`

For example, in Chapter 2, you entered the `pg` command to show file contents. CLEX searched the directories in your default command path for `pg`:

`.`	Not found in current directory.
`/sys/exe`	Not found in BiiN™ commands directory.
`/bin`	Found in BiiN™/UX commands directory.
`/usr/bin, /sys/etc/ex/tools`	Not searched.

Once you begin adding directories to your account, you may want to include them in your command path. For instance, most people create a personal directory for executable programs, for example `/users/joe/exe`, then put that directory in their command path.

To create a directory `/exe` in your home directory:

```
clex-> manage.directory
manage.directory=> create exe
manage.directory=> exit
```

To add the new directory `/users/joe/exe` to the command path, use `set.command_path`:

```
clex-> set.command_path (/sys/exe /bin /usr/bin \
CONTINUE CMD: /sys/etc/exe/tools /users/joe/exe .)
```

Abbreviation: `s.com`

Note that when you are entering a list of pathnames, you must enclose the list with parentheses, and you don't have to quote each pathname. Also, when entering a long command, you might want to use the backslash and continue the command on the next line. When you use the backslash, you will automatically be prompted to continue the command. Because the new command path replaces the old one, be sure to include all the directories you want to retain. Note: it's best to order your directories from most-used to least-used, with your current directory (dot) at the end of the list, to minimize search time.

To confirm that your command path is changed, use `list.command_path`:

```
clex-> list.command_path
("/sys/exe" "/bin" "/usr/bin" "/users/joe/exe" "/users/joe/exe" ".")
```

Note: if, during your session, you add an entry to any of the directories in your command path, be sure to issue a `set.command_path` with no arguments. This updates the system's list of the contents of the directories in your command path.

3.6 Examining BiiN™ CL Variables

In BiiN™ CL, a variable is simply a fixed name that holds a varying value.

CLEX and other system utilities use variables to allow you to customize the behavior of a program. For example, to set the prompt string or the number of last commands entered, you simply put your own value into the proper variable (`cli.prompt` or `cli.num_last_commands`, respectively). You can also create your own variables.

BiiN™ CL offers *variables* and *group variables*. BiiN™ CL variables have a single name with no dot, such as `$status`, and they are not saved on disk. BiiN™ CL group variables have a two-part name such as `cli.prompt`, and their values are saved on disk. The first part of the name is the *group* name; related variables are grouped together. Thus the variables in group `cli` affect the command-line interpreter, CLEX; the variables in group `logon` affect the `logon` process; the variables in group `print` affect the print spooler, and so on.

To see a list of the BiiN™ CL variables currently in effect for your account, use `list.variable`:

```
clex-> list.variable
STR   $TERM     "w5"
STR   $PATH     "/sys/exe:/bin:/usr/bin:/sys/etc/exe/tools:
                 /users/joe/exe:."
int   $status   0
ptr R $OEO      -- #directory object#
```

The display shows the following aspects of variables:

`STR`	Type; in this case, global string. Global variables are uppercase, local are lowercase. If the variable is read-only, an `R` appears after the type name (as in `$OEO`).
`$TERM`	Name.
`"w5"`	Value.

When you first start your account, you do not have your own personal values for group variables; when a value is needed, you use the system default. Later, when you want to change values, you can store your values in your personal directory `~/var_groups`.

To see a list of the system's default BiiN™ CL variable groups, list the entries in directory `/sys/var_groups`:

```
clex-> list.object /sys/var_groups
...
AMDS    cg      cobolg ...
ada     cli     debug  ...
```

Once you know the group name, you can list the default values of the variables in that group. For example, to list the variables in group `cli`:

```
clex-> list.variable cli.
STR   $cli.prompt       "Enter cmd => "
str   $cli.prompt       "clex-> "
STR   $cli.node         -- no value
STR   $cli.form_request         "on_request"
INT   $cli.num_last_cmds        30
BOO   $cli.verbose_history      false
STR   $cli.clex         "/sys/exe/clex"
```

See the *BiiN™ Command Language Executive Guide* for further information about variables.

3.7 Creating an Alias for a Command

To abbreviate long command names or frequently-used commands, you can write an *alias*. Just like an alias for a person, an alias for a command is an assumed name that is used instead of the original name.

To write a new alias (`ll`) for a long listing of directory entries, use `set.alias`:

```
clex-> set.alias ll "list.object :long"
```

Abbreviation: `set.al ll "list.object :long"`

You need to enclose the value in quotes (") because it contains a space.

Once your alias is created, any time you want a long listing you can use the alias:

```
clex-> ll
.                    02-09 08:47 directory   joe     umc       208

.:
.default_authority   02-09 08:49 authority_l joe     --c       208
...
```

It's best *not* to abbreviate the long command when you enter the value in quotes. This avoids ambiguous command names later. (Because the alias itself is short, you don't need the abbreviation anyway.)

`set.alias` sets aliases for the current job only. If you want your aliases to be set each time you logon, add appropriate `set.alias` commands to the logon startup file `~/startup/logon`.

3.8 Session 3 Summary

- `change.password` modifies a user's password.
- `list.user_profile` lists information about a specified user.
- `set.variable` assigns a value to a BiiN™ CL variable.
- The `~/startup/logon` file contains commands that are executed automatically when you logon.
- The `~/startup/clex` file contains commands that are executed automatically when you start CLEX.
- `list.command_path` displays your current command path.
- `set.command_path` assigns a new value to your command path.
- `set.alias` defines an alias name for a given string.

4 PRINTING FILES

This session shows you how to use the print spool queue for files.

After finishing this session, you'll know how to:

- Send a file to the print spooler
- Display the print spooler queue
- Remove a file from the print spooler

4.1 Sending a File to the Print Spooler

`print.file` sends a file to the print spooler.

To send the file `templ` to the default print spooler:

```
clex-> print.file templ /sys/spool_q
```

Abbreviation: `p.f`

4.2 Displaying the Print Spooler

`list.spool_file` lists the files in the print spooler.

To see `templ`'s place in the print spooler:

```
clex-> list.spool_file /sys/spool_q
user:                joe
file_ID:             1
file_size:           2117
file:                1/printing/...date...
printing_enabled:    true
files_auto_deleted:  true
copies:              1
term_msg             false
banner_page:         true
printers:            /vs/sys/_volset/dev...
```

Abbreviation: `l.sp_f`

4.3 Removing a File from the Print Spooler

`remove.spool_file` allows you to remove a file from the print spooler.

To remove the spooled file `templ`, enter its number (`File_ID`) in the queue:

```
clex-> remove.spool_file 1 /sys/spool_q
```

Abbreviation: `r.sp_f 1`

4.4 Section 4 Summary

- `print.file` queues one or more files for printing.
- `list.spool_file` lists information about files in a spool queue.
- `remove.spool_file` removes spooled files from a queue.

CONTROLLING JOBS 5

This session shows you and how to start and stop *jobs*. Generally, each command you enter runs as a job. You can run more than one command (job) at a time.

After finishing this session, you'll know how to:

- Run a job in the background
- List current jobs
- Stop a background job
- List previous commands
- Redo a previous command

5.1 Running a Job in the Background

You can run a job "in the background" using the BiiN™ CL option `&`. The command is started, and the prompt immediately returns so you can continue entering commands. This example uses the command `list.monitor_log` because it runs until you stop it.

To run `list.monitor_log` in the background:

```
clex-> list.monitor_log :block > temp6 &
clex: BACKGROUND JOB: list.monitor_log [list.monitor_log :block > temp6 &].
```

Abbreviation: `lis.mon :b > temp6 &`

5.2 Listing Current Jobs

You can display your background jobs (and other jobs) with `list.job`.

To list your current jobs:

```
clex-> list.job

SESSION  joe on tty017, CREATED 09May, 14:04:48.17
(1) exec   list.monitor_log [list.monitor_log :block > temp6 &]
...
```

Abbreviation: `lis.job`

5.3 Stopping a Background Job

You can stop a background job with `kill.job`.

To stop the background job `list.monitor_log`:

```
clex-> kill.job 1
clex: KILL event signalled to job 'list.monitor_log [list.monitor_log :block
      > temp6 &].'
clex-> list.monitor_log - E: Terminated by event 5.
```

Notice that the message window shows the completion of the background job, with exit status `'2'` (error).

5.4 Listing Previous Commands

CLEX remembers the 30 previous commands you typed. You can list these and re-do a previous command.

To list your previous commands, use `list.last_commands`:

```
clex-> list.last_commands
...
more?(<blank> | <lf> | d | q) <SPACEBAR>
...
(46) list.monitor_log :block > temp6 &
(47) list.job
(48) kill.job 1
```

Abbreviation: `lis.las`

Note: you can change the number of remembered commands by changing the value of the variable `$cli.num_last_cmds`.

5.5 Redoing a Previous Command

To redo a previous command, use `redo.last_commands`. You can specify which previous command either by its number or its command name.

To redo the last `list.job`:

```
clex-> redo.last_commands 47
list.job

SESSION joe on tty017, CREATED...
(1) exec   Session_Server
(-) exec   Logon_CLEX
```

Abbreviation: `re.las 47`

5.6 Section 5 Summary

- The & BiiN™ CL option runs a job in the background.

- `list.job` lists currently running jobs.
- `kill.job` stops the specified job.
- `list.last_commands` lists the previous commands entered.
- `redo.last_commands` reexecutes a previous command.

WORKING WITH WINDOWS 6

A *window* is an area of your terminal screen that acts like an independent "subterminal".

Take a minute to study the title bars on your two windows. Each window's *title bar* tells you the window's number and command. The default windows after login are window 1 for reading messages and window 2 for entering commands to `clex`. Windows do not overlap.

After finishing this session, you'll know how to:

- Open a new window
- Change between windows
- Resize a window
- Get help with window commands
- Close the new window

If Your Terminal Beeps. If your terminal beeps when you enter a window command, it's an error message. For example, you'll get a beep for a typing mistake, an unknown window number, and so on.

6.1 Opening a New Window

The `::window` option opens a new window for the command being entered.

To open a second CLEX window:

```
clex-> clex ::window
```

Abbreviation: `cl ::w`

The new window, number 3, opens below the old one, with the command `clex` in the title bar. This new `clex` is a *non-logon* `clex`, so it has a different prompt: `Enter cmd =>`. Note that the previous windows are resized. Window 3 becomes the *current window*, the one in which you enter commands (note the `F` in the title bar, for *focus*, in the following display).

```
 1  Message Window
clex: Job completed, status '2': list.monitor_log [list.monitor_log :block
       > temp6 &]
 2  Initial program: /vs/sys_volset/exe/clex
clex: BACKGROUND JOB: clex [clex ::window].
clex->
 3  clex [clex ::window]                                              F




CLEX Command Language Executive,   VERSION:  nn
Enter cmd =>
```

To confirm that you can enter commands in the new window just like the old one, try `get.time` and `list.object`:

```
clex-> get.time
1988-04-01 16:06:42.78
clex-> list.object
.default_authority      exe             temp6
.logon_script           startup         var_groups
.mail_AL                temp1
```

6.2 Changing Windows

The command **<Ctrl-T>**2 changes to window 2. (In general, **<Ctrl-T>***n* changes to window *n*.)

You do not enter a <RETURN> after window commands. If you do enter a <RETURN> out of habit, the <RETURN> will be taken as input. **Window commands are not echoed on the screen.**

To change from your current window (3) to the logon `clex` window (2):

```
Enter cmd => <Ctrl-T>2
```

The cursor is now at the prompt in window 2, and window 2 is the current window which receives your commands.

Experiment with **<Ctrl-T>** and a window number until you are comfortable with changing windows.

6.3 Resizing a Window

You can change the size of a window with the <Ctrl-T><Shift-L> and <Ctrl-T>s commands. <Ctrl-T><Shift-L> makes a window as large as possible. <Ctrl-T>s makes a window smaller by a given number of lines.

First go to window 3:

```
Enter cmd => <Ctrl-T>3
```

Use <Ctrl-T><Shift-L> to make window 3 as large as possible:

```
Enter cmd => <Ctrl-T><Shift-L>
```

The other windows (1 and 2) only show their title bars, and window 3 occupies all the other lines on the screen (see the following display).

```
1  Message Window
2  Initial program: /vs/sys_volset/exe/clex
3  clex [clex ::window                                              F

CLEX Command Language Executive,   VERSION:  nn
Enter cmd => get.time
1988-04-01 16:06:42.78
Enter cmd => list.object
.default_authority    exe         startup    temp6
.mail_authority       personal    temp1      var_groups
```

You can make a window smaller with <Ctrl-T>s. <Ctrl-T>s takes a number from 0 through 9.

For example, window 3 is now as large as possible. Suppose you want to make it smaller so as to leave more room for window 2. To make window 3 smaller by 9 lines, so that it takes up about half the screen, enter:

```
Enter cmd => <Ctrl-T>s9
```

There must be no spaces in the window command.

Windows 1 and 2 are resized larger.

6.4 Getting Help with Window Commands

You can show a list of the window commands with <Ctrl-T>?:

```
Enter cmd => <Ctrl-T><?>
```

6.5 Closing a New Window

A window goes away when the command that started it ends.

To close window 3, use the `clex` command `exit`, which exits from a non-logon `clex`:

```
Enter cmd => exit
```

The new window disappears and windows 1 and 2 resume their previous places.

6.6 Session 6 Summary

- `::window` is a BiiN™ CL option to open a new window.
- <Ctrl-T> prefixes window commands.
- <Ctrl-T>? shows the list of window commands.
- `exit` exits from a non-logon `clex`.

7 PROTECTING FILES AND OTHER OBJECTS

Each directory in the BiiN™ system is protected from unauthorized access; that is, you control who can access your home directory. Directories, like other BiiN™ objects, are protected with an *authority list.* The authority list specifies which IDs can access the directory, and with what access rights.

After finishing this session, you'll know how to:

- List the default protection for your home directory
- Make a directory private
- Add modify rights to a file
- Change a directory's default protection
- Examine your ID list

7.1 Listing Default Protection for Your Directories

Each user account has an initial *default authority list*, in `.default_authority` in your home directory. This list specifies the protection that is automatically assigned to your home directory and all the objects you will create in it, such as files and new directories.

To see the contents of your default authority list, use the `list` runtime command of `manage.authority` with the `suppress` argument:

```
clex-> manage.authority
manage.authority => list ~/.default_authority :suppress
Default authority list: ///org/dom/vs/users/joe/.default_authority
    umc      ///org/dom/id/joe
    u--      ///org/dom/id/sysgroup
    u--      ///org/dom/id/world
```

The `list` runtime command displays the object's protecting authority list.

If the object is itself an authority list and you want to see the contents, use the `:suppress` argument (otherwise you'll see the authority list that protects the authority list).

Your system administrator sets the initial authority list. The first ID in the list is your user ID (logon name). Your user ID has `use`, `modify`, and `control` rights. For a directory, this means you can do any directory operations on it.

The second and third IDs (`sysgroup` and `world`) give everyone else `use` rights. These IDs can list your directory entries but cannot do anything else.

(A note about the IDs: a pathname with three leading slashes `///` is the system's full pathname for an ID.)

The authority list that protects your home directory is the *same one* that is used to automatically protect all the new objects you will create in your directory structure.

To exit from `manage.authority` and return to CLEX:

```
manage.authority => exit
clex->
```

7.2 Making a Directory Private

By default, all users are allowed to list the contents of your directories. To make a directory private (only you can list it), you need to create a new authority list that has only your access rights in it, then assign the new authority list to that directory.

To create a new authority list `private_auth`, invoke `manage.authority` and use the runtime command `create`:

```
clex-> manage.authority
manage.authority => create private_auth (joe umc)
```

You enter only your ID, with all rights, as the *protection set* (list of <ID, rights> pairs).

To associate the new, private authority list with the directory `personal`, use the runtime command `set.object_authority`:

```
manage.authority => set.object_authority personal private_auth
```

Abbreviation: `set.ob`

The directory `personal` is now protected with the authority list `private_auth`.

7.3 Confirming the New Authority List

To confirm the new, private authority list for the directory `personal`, use the runtime command `list`:

```
manage.authority => list personal
 umc       ///org/dom/id/joe
```

Your ID is the only one in the new directory's authority list.

Now that the private authority list is created, set, and confirmed, exit from `manage.authority`:

```
manage.authority => exit
```

7.4 Adding Group Modify Rights to a File

You can also make a file or other object more public (allow others to modify.) For example, suppose you wanted to allow others to write to a file. You create a new authority list, allowing modify writes for the group ID, then assign the new authority list to the file.

For example, use the `list` runtime command to see the authority list protecting `templ`:

```
clex-> manage.authority
manage.authority=> list templ
Protecting authority list: ///org/dom/vs/users/joe/.default_authority
    umc      ///org/dom/id/joe
    u--      ///org/dom/id/sysgroup
    u--      ///org/dom/id/world
```

To add group modify rights to the file `templ`, create a new authority list named `group_m_auth` and assign it to `templ`:

```
manage.authority=> create group_m_auth (joe umc sysgroup um world u)
manage.authority=> set.object_authority templ group_m_auth
```

(Notice that the other IDs are still included with their rights unchanged. If you did not include the IDs, then `joe` and `world` would not be able to access `templ`.)

You can confirm `templ`'s new authority list:

```
manage.authority=> list templ
Protecting authority list: ///org/dom/vs/users/joe/group_m_auth
    umc      ///org/dom/id/joe
    um-      ///org/dom/id/sysgroup
    u--      ///org/dom/id/world
```

Finally, you can exit from `manage.authority`:

```
manage.authority=> exit
```

7.5 Changing a Directory's Default Protection

Another way to make a directory private is to change its *default authority list.* When you create a new entry in a directory, the entry is automatically protected by the default authority list unless you specify a different authority list. So to make entries readable only by you, you could change the default authority list of your private directory. Then whenever you create an entry in that directory, the entry is readable only by you.

```
clex-> manage.directory
manage.directory=> set.default_authority personal private_auth
manage.directory=> exit
```

Note the differences in protecting a directory with its own authority list or its default authority list. If only the directory's default authority list is private, then others will be able to see the name of the private directory in your home directory, but will not be able to list entries. When the directory's own authority list is private, no one but you will be able to even list the name of the directory in your home directory.

7.6 Examining Your ID List

You can access any system command (or other object), if you have an ID that matches one in the object's authority list. You may have more than one ID at a time. Your *ID list* shows the IDs under which you are allowed to access commands and objects (that is, who you can represent—a member of the finance department or a database user, for example).

Your current ID list is always available in the BiiN™ CL group variable `$pglob.id_list`. To see your current ID list:

```
clex-> list.variable pglob.id_list
LIS R $pglob.id_list    ("///org/dom/id/joe" "///org/dom/id/sysgroup"
                         "///org/dom/id/world")
```

Abbreviation: `lis.var`

In this example, the first entry is Joe's identity, the second entry is a group ID, and the third entry is everyone on the system. Hence, Joe can run programs/scripts and access directories/files (among other things) as himself, as a member of the `sysgroup` group, or as `world`.

However, his access rights will vary depending upon a specific object's authority list. For example, one object may allow use and modify rights for the ID `joe`, while another may have no rights listed for `joe` at all, in which case `joe` cannot access the object.

`pglob` is short for *process globals*. The `pglob` variable group contains values for your processes that are currently executing (a job contains at least one process).

7.7 Session 7 Summary

- `manage.authority` creates, assigns, and lists the contents of authority lists.
- `manage.directory` lists and assigns default authority lists.
- `list.variable pglob.id_list` shows your current ID list.

7.8 So You've Finished

The goal of these sessions has been to give you experience entering the commands you'll need to do basic daily work on a BiiN™ system.

You're not expected to remember everything you did here. As you become more familiar with the BiiN™ system, review the examples in this manual, practice, and experiment. Remember that the complete description for BiiN™ CL commands is in the *BiiN™ Systems Commands Reference Manual.*

Several topics were intentionally left out of these sessions. For further learning on the following topics, here are the manuals you will need:

Text editing	The BiiN™ text editor is Emacs. The manual *BiiN™ Systems Emacs User's Guide* provides tutorials on text editing.
CLEX command interpreter	(CLEX) has many more features than were presented here, including flow control and script writing. The manual *BiiN™ Command Language Executive Guide* provides tutorials on the command language BiiN™ CL and the program that interprets the language, CLEX.
Compiling	Refer to the manual for your preferred language for tutorials on using the compiler for your language. For example, the COBOL manual is *BiiN™ COBOL Programming Manual.*
Linking	The BiiN™ Systems Linker manual is *BiiN™ Systems Linker Guide.*

COMMAND QUICK REFERENCE A

This appendix summarizes the commands presented in this manual, the help commands, and syntax notation.

A.1 Summary of Commands

The following sections summarize the commands and examples presented in this manual, in the following format:

`command` Description of command.

`command` *arg1 arg2*
(General form of command as it is commonly entered; may not appear if obvious. Italics represent arguments that are to be replaced by your actual values.)

```
command arg1 arg2
```
(Example as presented in this manual; may not appear if obvious.)

A.1.1 Files and Directories

`copy.object` Copies an object from one pathname to another pathname.

`copy.object` *orig copy*
```
copy.object temp1 temp2
```

`rename.object`
Renames an object's directory entry.

`rename.object` *old new*
```
rename.object temp2 temp3
```

`remove.object`
Removes an object's directory entry.

`remove.object` *entry*
```
remove.object temp3
```

`manage.directory`
Creates, lists, and sets authorization for, directories.

```
manage.directory
  => create directory
  => exit

manage.directory
  => create personal
  => exit
```

`list.current_directory`
Lists the current directory's pathname.

`set.current_directory`
Sets the current directory's pathname.

`set.current_directory` *directory*
`set.current_directory personal`

`>` BiiN™ CL option to redirect output.

command-output `>` *file*
`list.object /exe > templ`

A.1.2 Logon, Logoff, Help

Logon Service — Allows a user to logon to the system.

<RETURN>
=> *logon-name*
=> *password*

<RETURN>
=> `joe`
=> `newuser`

`logoff` — Terminates a logon CLEX.

<BACKSPACE> — Erases character to the left of the cursor.

<Ctrl-C> — Cancels current input (returns to `clex->` prompt).

`get.time` — Gets the system time.

`list.object` — Lists entries in a directory.

`list.status user :admin`
Lists processes, jobs, sessions, and active users.

`?` — Question mark displays syntax or description help for a command. Single question mark usually displays syntax only.

command `?` or
command `??`

```
list.spool_rank ?
list.spool_rank ??
```

A.1.3 User Account

`change.password`
Changes a password for a user or other ID.

```
change.password
  => old
  => new
  => new

change.password
```

```
=> newuser
=> sesame
=> sesame
```

`list.user_profile`
Lists a user's profile, protection set, and default authority list.

`set.variable` Sets a list of BiiN™ CL variables to a value.

```
set.variable name "value"
set.variable cli.prompt "clex-> "
```

`list.command_path`
Lists the pathnames in the current command path.

`set.command_path`
Sets the command path.

```
set.command_path (dir1 dir2 ...)
set.command_path (/sys/exe /bin /users/joe/exe .)
```

`list.variable`
Lists the types, modes, names, and values of BiiN™ CL variables.

```
list.variable   or
list.variable name

list.variable
list.variable cli.
```

`set.alias` Creates, or assigns a value to, an alias.

```
set.alias  name   "value"
set.alias ll "list.object :long"
```

A.1.4 Printing Files

`print.file` Queues one or more files for printing.

```
print.file file
print.file temp1
```

`list.spool_file`
Lists names of spooled files for one or more users.

`remove.spool_file`
Removes spooled files from a queue.

```
remove.spool_file file-number
remove.spool_file N
```

A.1.5 Controlling Jobs

`&` BiiN™ CL option to run a background job.

```
command &
list.monitor_log :block > temp6 &
```

`list.job` Lists the jobs in the user's sessions.

`kill.job` Kills a job.

```
kill.job number or name
kill.job 1
```

`list.last_commands`
Lists a user's last few commands.

`redo.last_commands`
Redoes one or more previously entered commands.

```
redo.last_commands number or name
redo.last_commands 22
```

A.1.6 Using Windows

`::window` BiiN™ CL option to open a new window to execute a command. Window closes when command execution finishes.

```
command ::window
clex ::window
```

[exit] CLEX command to exit (closes window).

<Ctrl-T> Window command prefix. **<Ctrl-T>***n* changes to window *n*.

<Ctrl-T>*number*

<Ctrl-T>2
<Ctrl-T>3

<Ctrl-T><Shift-L>
Makes current window as large as possible.

<Ctrl-T>s9 Makes current window smaller by 9 lines.

<Ctrl-T>? Displays list of window commands.

A.1.7 Protecting Objects

`manage.authority`
Manages authority lists (show contents of an authority list).

```
manage.authority
  => list aut-list :suppress
  => exit

manage.authority
  => list ~/.default_authority :suppress
  => exit
```

`manage.authority`
Manages authority lists (create and assign an authority list).

```
manage.authority
  => create  aut-list  (id1 rights1 id2 rights2 ...)
  => set.object_authority object aut-list
  => exit
```

```
manage.authority
  => create private_auth (joe umc)
  => set.object_authority personal private_auth
  => exit
```

`manage.authority`

Manages authority lists (display protecting authority list).

```
manage.authority
  => list object
  => exit

manage.authority
  => list personal
  => exit
```

`manage.directory`

Manages directories (set default authority list).

```
manage.directory
  => set.default_authority directory aut-list
  => exit

manage.directory
  => set.default_authority personal private_auth
  => exit
```

`list.variable pglob.id_list`

Lists value of BiiN™ CL variable (in this case, ID list).

The help commands ? and ?? can show syntax, descriptions, or lists of CLEX commands (Table A-1).

Table A-1. Help Commands

Task	Format	Examples
Command syntax	*command* ?	clex-> list.user_profile ? lup [:user=<symbolic_name_list(0..1_000_000(0..128))>:=$user.name] [:long=<boolean>:= false] CONTINUE CMD:
Command syntax + description	*command* ??	clex-> list.user_profile ?? lup [:user=<symbolic_name_list(0..1_000_000(0..128))>:=$user.name] [:long=<boolean>:= false] -- Lists user profiles for one or... -- A complete profile ... CONTINUE CMD:
Argument syntax + description	*argument*=?	clex-> list.user_profile :user=? [:user=<symbolic_name_list(0..1_000_000(0..128))>:=$user.name] -- One or more user's logon names. -- is to be reported. If null, lists -- all users. CONTINUE CMD:
Command syntax + arg description	*argument*=??	clex-> list.user_profile :user=?? list.user_profile [:user=<symbolic_name> := user.name] [:long=<boolean> := false] CONTINUE CMD:
List of runtime commands	*prompt* ?	manage.directory=> ? PROGRAM COMMANDS: exit create ... clex-> ? CLEX COMMANDS: {suspend|resume...}.job set.event_action ...
List of BiiN™ CL builtin commands	*prompt* ??	manage.directory=> ?? COMMON BUILTIN COMMANDS: {create|set...}.variable {set|list|remove}.alias ... clex-> ?? COMMON BUILTIN COMMANDS: {create|set...}.variable {set|list|remove}.alias ...

UNIX AND BiiN™ COMMANDS B

This appendix shows which BiiN™ CL commands are equivalent to UNIX commands, to help readers familiar with a UNIX environment assimilate BiiN™ CL quickly.

Table B-1. UNIX Commands and BiiN™ Equivalents

UNIX Command Equivalent	CL Command
alias (C shell)	set.alias
cat .login	cat ~/startup/logon
cd	set.current_directory
chmod	manage.authority
cp	copy.object
csh, sh	clex
date	get.time
emacs	emacs
history (C shell)	list.last_commands
jobs (C shell)	list.job
kill	kill.job
logout, ^D	logoff
lp, lpr	print.file
lpq, lprm	list.spool_file, remove.spool_file
ls	list.object
man	?, ??
mkdir	manage.directory => create
mv	rename.object
passwd	change.password
pg, cat	pg, cat
pwd	list.current_directory
rm	remove.object
!n	redo.last_commands
&, >, <	&, >, <

SUMMARY OF WINDOW COMMANDS C

Table C-1 describes the BiiN™ window commands.

Table C-1. Window Commands

Command	Key Sequence	Description
Help	<Ctrl-T><?>	Lists these window commands.
Change focus to window *n*	<Ctrl-T><*n*>	Changes the input focus to window *n*. Window numbers range from *1* to *9*. Changing a window's screen position does not change its number.
Hide window	<Ctrl-T><h>	Removes the window from the screen, but does not destroy it or prohibit operations on it (for example, the application can write to it). Hiding a window does not change its number. To redisplay a hidden window, use *Change focus to window n*. All windows except the last one can be hidden.
List hidden windows	<Ctrl-T><w>	Displays the list of currently hidden windows.
Resize window larger	<Ctrl-T><l><*n*>	Increments the size of the window by *n* rows, if possible. Otherwise, makes the window as large as possible.
Resize window smaller	<Ctrl-T><s><*n*>	Decrements the size of the window by *n* rows, if possible. Otherwise, makes the window as small as possible (1 row plus title bar).
Make window as large as possible	<Ctrl-T><Shift-L>	Makes the window as large as possible, subject to the restrictions imposed by the frame buffer and other windows.
Make window as small as possible	<Ctrl-T><Shift-S>	Reduces the window to the minimum size (0 rows plus title bar).
Set desired window size	<Ctrl-T><x>	Sets the desired window size. The application may set the desired window size when the window is created; this command allows the user to reset it. For example, CLEX may set the desired window size to 20 lines, but the user can make it smaller by changing focus to the CLEX window, resizing it smaller, and then issuing this command, which sets the desired window size to the current (smaller) size.
Move window to top of screen	<Ctrl-T><t>	Moves the current window to the top of the screen. Moves other windows down, if necessary.
Move window to bottom of screen	<Ctrl-T><b>	Moves the current window to the bottom of the screen. Moves other windows up, if necessary.
Scroll to top	<Ctrl-T><a>	Pans the view to the top of the frame buffer.
Scroll to bottom	<Ctrl-T><z>	Pans the view to the bottom of the frame buffer.
Scroll up page	<Ctrl-T><Shift-K>	Pans the view up one page on the frame buffer (or as far as possible). A page equals the size of the view.
Scroll down page	<Ctrl-T><Shift-J>	Pans the view down one page on the frame buffer (or as far as possible). A page equals the size of the view.
Scroll up half page	<Ctrl-T><u>	Pans the view up a half page on the frame buffer (or as far as possible). A page equals the size of the view.
Scroll down half page	<Ctrl-T><d>	Pans the view down a half page on the frame buffer (or as far as possible). A page equals the size of the view.
Scroll up row	<Ctrl-T><k>	Pans the view up one row on the frame buffer, if possible.
Scroll down row	<Ctrl-T><j>	Pans the view down one row on the frame buffer, if possible.
Move view to cursor	<Ctrl-T><v>	Moves the view up or down in the frame buffer, as needed, until the cursor is just inside the view. (For use when the cursor is outside the view.)
Redraw screen	<Ctrl-T><r>	Redraws the terminal screen.
Request closing of window	<Ctrl-T><c>	Generates a `close_requested` input event for the current window. The application is responsible for taking appropriate action. This command does not itself close the window.
Start menu interaction	<Ctrl-T><m>	Invokes pull-down menu.

ROADMAP TO BiiN™ DOCUMENTATION D

There are many pages in the BiiN™ document set. However, with the help of the document set roadmap in Figure D-1 you should be able to find the information you need.

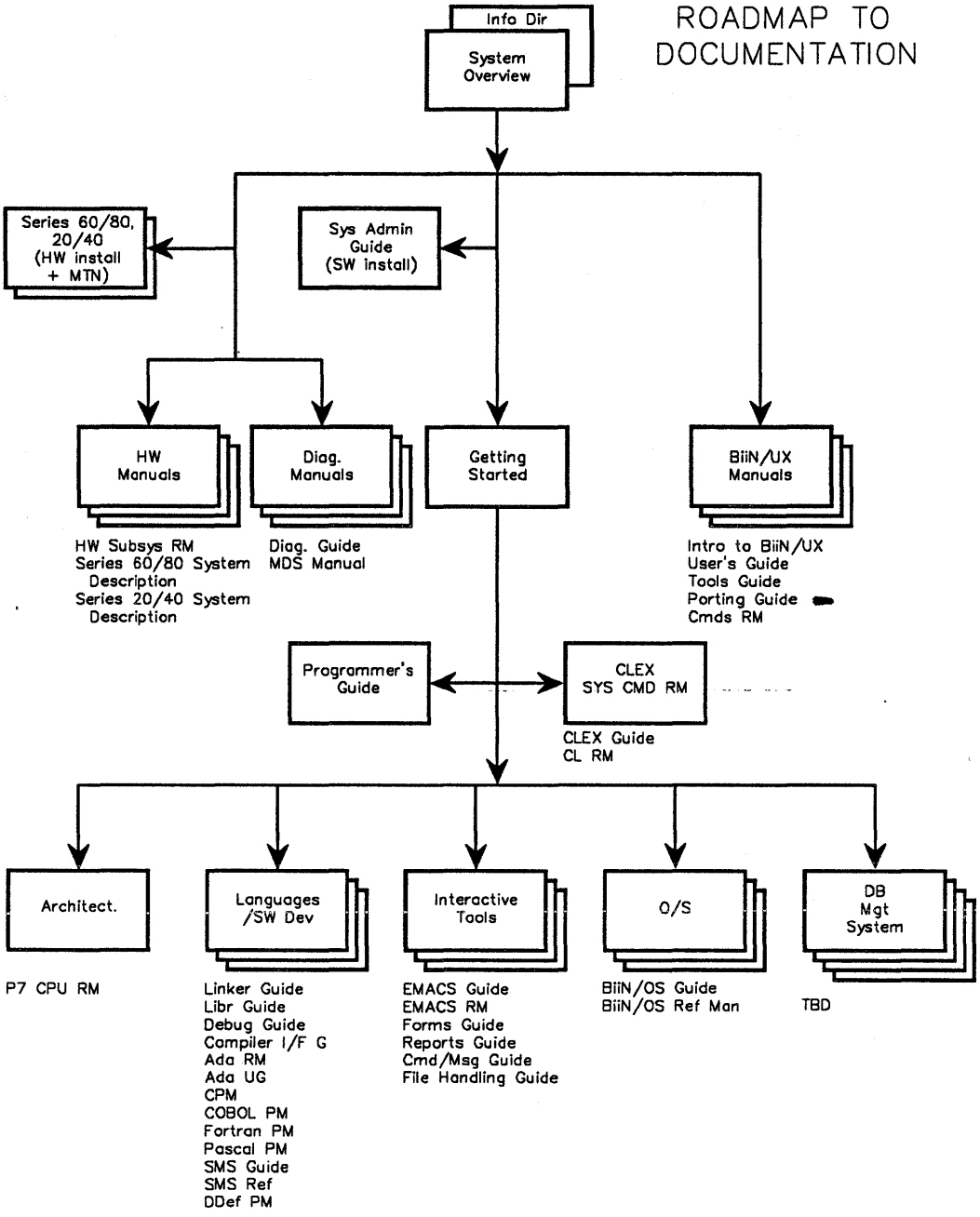


Figure D-1. Roadmap to BiiN™ Documentation

INDEX

L

M

N

O

P

R

S

T

U

W