Installing and Configuring

Interactive COBOL

on Linux

Revision 5.40

No. 011-00402-26

May 2020

LICENSE AGREEMENT

Carefully read the following terms and conditions. **Use of this product constitutes your acceptance of these terms and conditions and your agreement to abide by them.**

You, the purchaser, are granted a non-exclusive license to use this software under the terms stated in this agreement. The program and its documentation are copyrighted and may not be copied or reproduced in any part, in any form, for any purpose, except according to the terms stated in this agreement.

You may:

1. use the software for up to the number of active users for which the software was purchased.

2. use the software provided a valid license is installed for the required number of active users to be supported at any one time.

3. copy the software into any machine readable form for backup purposes.

4. transfer the software from one computer to another.

5. assign or transfer the software and license to another party if the other party agrees to all the terms and conditions of this agreement. Once the transfer is complete you must destroy any copies of the software not transferred.

6. rent, sublicense, or lease the software and license if the user agrees to all the terms and conditions of this agreement.

7. not alter, modify, or adapt the software itself, including, but not limited to, translating, decompiling, or disassembling.

8. copy or reproduce the documentation for purposes of using a valid license.

This license and your right to use the software automatically terminate if you fail to comply with any provision of this License Agreement. You agree upon such termination to destroy the software and license.

Limited Warranty

Envyr Corporation warrants that (a) the software will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt; and (b) any hardware accompanying the software will be free from defects in materials and construction under normal use and service for a period of one (1) year from the date of receipt. Any implied warranties on the software and hardware are limited to ninety (90) days and one (1) year respectively. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Envyr Corporation's entire liability and your exclusive remedy shall be, at Envyr Corporation's option, either (a) return the license fee or (b) repair or replacement of the software or hardware that does not meet the above Limited Warranty and which is returned to the original vendor with a copy of the receipt. This Limited Warranty is void if failure of the software or hardware has resulted from accident, abuse, or misapplication.

In no event shall Envyr Corporation or its suppliers be liable for any damages at all, including, but without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss, arising out of the use of or inability to use this software or hardware, even if Envyr Corporation has been advised of the possibility of such damages.

Envyr Corporation
92 Cornerstone Dr Ste 143
Cary, NC 27519-8404
USA

www.icobol.com

NOTICE

This manual has been prepared for use only with the Interactive COBOL product by prospective customers or valid licensees. The information in this manual is preliminary and subject to change without prior notice.

In no event shall the seller be liable for any incidental, indirect, special or consequential damages at all (including but not limited to lost profits) arising out of or related to this document or the information contained in it even if the writers have been advised, knew or should have known of the possibility of such damage.

Revision History:

Release	2.00	- March 1994	
Release	2.20	- August 1996	
Release	2.30	- July 1997	
Release	2.40	- June 1998	
Release	3.00	- August 2000	
Release	3.10	- April 2001	
Release	3.20	- April 2002	
Release	3.30	- February 2003	
Release	3.40	- March 2004	
Release	3.64	- January 2008	
Release	3.65	- March 2008	
Release	4.00	- October 2008	
Release	4.01	- December 2008	
Release	4.04	- February 2009	
Release	4.07	- June 2009	
Release	4.10	- August 2009	
Release	4.20	- December 2009	
Release	4.40	- June 2010	
Release	4.50	- April 2011	
Release	4.54	- November 2011	
Release	4.70	- August 2012	
Release	5.00	- November 2014	(Manual name change from UNIX to Linux)
Release	5.02	- December 2014	
Release	5.10	- October 2015	
Release	5.20	- May 2016	
Release	5.40	- May 2020	

Effective with:
Interactive COBOL Revision 5.40

TRADEMARKS

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF SCREENS

PREFACE

Scope

This manual provides the information needed to install and configure the Interactive COBOL product on Linux.

The complete documentation for Interactive COBOL includes the following manuals:

Installing and Configuring Interactive COBOL on Linux (011-00402)
Installing and Configuring Interactive COBOL on Windows® (011-00403)
Provides the appropriate sections necessary to properly install and configure Interactive COBOL in the given environment.

Interactive COBOL Utilities Manual (011-00300)
Provides a simple guide to all the user visible utilities.

Interactive COBOL Language Reference & Developer's Guide (011-00100)
Provides the complete COBOL syntax supported by Interactive COBOL. Shows how to use the development tools including the compiler and IDE. It also explains how the COBOL runtime works including its internal system calls, builtins, and how to program across the multiple environments supported by Interactive COBOL.

sp2 panel Editor
Provides how to develop and use the sp2 User Interface (GUI) Development System (ICSP2).

COBOL FormPrint
Provides how to use the FormPrinter Editor (ICQPRW) to setup printers.

Terms

This document will use several terms which it will define as generic names to describe several individual products.

ICOBOL refers to any of the Interactive COBOL products unless otherwise stated.

Linux refers to all supported flavors of Linux unless specifically stated. See the readme for specific operating environments/flavors.

DG refers to Data General Corporation.

I. INTRODUCTION

A. Overview

The Interactive COBOL product set is composed of several products: The Interactive COBOL Runtime System, the Interactive COBOL Development System, the Interactive COBOL GUI Development System, and the FormPrint Editor.

The Interactive COBOL Runtime is the product which runs standard Interactive COBOL programs along with the needed utilities. The Runtime System is all that is needed to distribute Interactive COBOL applications that do not need development capabilities.

The Interactive COBOL Development System adds the Interactive COBOL compiler to the Runtime System. The Development System allows COBOL applications to be built and debugged. On Windows, an integrated development environment (ICIDE) is also included. Programs compiled on one system can be run on any system.

The Interactive COBOL GUI Development Systems (ICSP2) provides the sp2 Panel Editor and other needed files to allow graphical screens to be developed for deployment on Windows. The runtime support for "panels" created with ICSP2 is provided in the Windows runtime release. In addition, gui support for sp2 is provided in the Linux runtimes when communicating with a ThinClient client on Windows.

COBOL FormPrint (ICQPRW) is a tool used by COBOL programmers to develop and maintain contemporary print forms for their programs. It allows the complexities of such forms to be somewhat isolated from a COBOL program, yet still enables a COBOL programmer to have close control over the form directly from a program using standard COBOL code. The runtime support for "panels" created with ICQPRW is provided in the Windows runtime release. In addition, gui support for FormPrint is provided in the Linux runtimes when communicating with a ThinClient client on Windows.

The Interactive COBOL Client/Server facility (ICNETD) is a server that provides the server side support for both the thick-client and thin-client client/server offerings across a network between Linux and Windows machines. It provides thick-client support with the ICIOS server and ThinClient support with the ICRUNRS server, and remote SQL support with the ICSQLS server.

The cgiCOBOL runtime (ICRUNCGI) enables the use of **ICOBOL** programs as Common Gateway Interface (CGI) scripts, initiated by a web server in response to a browser request. CgiCOBOL programs can be written using standard ANSI COBOL syntax.

This manual will discuss:

1) how to install Interactive COBOL,
2) how to configure the runtime environment, and
3) how to start COBOL programs.

Interactive COBOL software is available for Linux (Intel).

This system employs a serial protection device that supplies a unique serial number to which a license is keyed or a MAC-based license based on the ethernet card. The actual license gives the maximum number of active terminals to be logged on simultaneously to the runtime along with any additional products licensed for use by this installation. Licenses can be shared over a network with other platforms including Windows to Linux or Linux to Windows.

B. How to Read this Manual

Begin with the Installation Chapter (Chapter II) which describes how to install Interactive COBOL on Linux, required changes to the operating environment, and tips and techniques to get the most from your system.

Scan the readme file(s) for any new information that is not in the manuals.

Proceed with the Licensing (ICPERMIT) Chapter (Chapter III) which describes how to start and stop the Interactive COBOL License Server process (ICPERMIT).

Continue the installation process by reading the ICCONFIG Chapter (Chapter IV), which describes how to use the Interactive COBOL configuration utility to prepare a configuration file with which to run Interactive COBOL.

Proceed with the ICEXEC Chapter (Chapter V) which describes how to start and stop the Interactive COBOL Executive process (ICEXEC) which reads the configuration file.

Proceed with the Starting ICRUN Chapter (Chapter VI) which describes how to start and stop an Interactive COBOL runtime process.

Chapter VII describes ThinClient support.

Chapter VIII describes some tuning tips for Interactive COBOL.

Chapter IX describes the default ICTERM definitions.

The APPENDIX section contains various information, including an ASCII Chart, an RS-232C discussion, support suggestions, etc., to be used as reference material.

C. Operating Environment

C.1. General Concepts

The Interactive COBOL system has been designed to provide an application operating environment that works as consistently as possible among several different operating system environments. This consistency is expressed in a few key concepts that have their roots in the MS-DOS and UNIX operating systems. If you are using one of these two operating systems, the concepts may already be familiar to you.

The first concept is that programs communicate with their operating environment through three input/output streams or files: standard input (stdin), standard output (stdout), and standard error (stderr). Programs can read data to be processed from stdin, process it in some way, and write the results to stdout. They report errors to stderr. By default, most systems connect stdin to the console keyboard and both stdout and stderr to the console display.

Many utilities, especially in the COBOL environment, must process complex data files that do not fit this simple model and so they do not often use stdin for the data to process. However, the stdout and stderr files are still very useful. They allow the utility to logically separate error reporting from reporting the results of processing. For example, the ICSTAT utility reports statistics about the ICISAM files. It reports these statistics to stdout. If an error occurs, (e.g., one of the command arguments does not exist), a message is reported to stderr.

The second concept is the ability to redirect i/o files from the default files to another file or device. Linux systems provide a very simple way to redirect these standard files in the command processor by using the special characters `<' and `>'. When stdout is redirected to a file, it provides a simple mechanism to capture the output of a utility. See your operating system shell documentation for more on this concept.

The third major concept is the ability to customize the operation of specific programs by setting information in items called Environment Variables. Environment variables have a name and a value like program variables or data items. The difference is that these variables are managed by the command processor. The utility programs can ask the operating system whether a particular environment variable is set or not, and what its value is. They are most often used to set default operating options, or the locations of important files. For example, all Interactive COBOL command-line programs look for the environment variable ICROOT as the base directory for finding the system directories. ICCONFIGDIR is also used to provide customized system files for help, messages, print, and term entries. Linux provides environment variables support.

Environment variables are maintained in the command processor (or shell). Environment variables are setup with a command like:

```
ICROOT=/usr/icobol.XXX
export ICROOT
```

C.2. Directory Structure

The Interactive COBOL software is installed in a directory with the name icobol.*nnn*, where *nnn* corresponds to the revision level. For example, Interactive COBOL Revision 5.40 will be in a directory named icobol.540. This directory can be installed wherever is most appropriate or convenient for your system and should be included on your PATH.

The main directory contains all of the command-line programs, the readme file(s), and supplied COBOL executable programs. One of its subdirectories is called *help*. The help subdirectory contains help (.hf) files for all the command-line programs defined as <command>.hf. There may be additional directories with other miscellaneous files.

Main Directory	Sub-Directories	Description
icobol.<rev>		Main executables, .so, and needed files
	docs	All documentation, readme files
	examples	Various examples
	cgi	Cgiruntime, scripts, examples
	config	various configuration files, .pti, messages
	icodbc	sample ICISAM odbc files
	print	various pdf sample backgrounds
	programs	Examples, login, sp2logon, isqltest, ...
	scripts	Various script files
	terminfo	Various terminfo source files
	help	Help files (.hf)
	icnet	Server surrogate files
	install	Various install scripts
	x86	On a 64-bit os this holds the matching 32-bit executables

Installs previous to 5.00 used a directory called cobolXXX.

Installs previous to 4.70 had a print sub-directory for printer translation (.pti) files and background .pdf files and a term sub-directory for term description (.tdi) files. The default versions of .pti and .tdi files are now built into the runtimes and any customized file(s) should be stored in a directory that is sought with the ICCONFIGDIR environment entry.

Command-line programs require the corresponding help file to be available in order to display their help text. If it is not available, an error message will be displayed that it could not find the help file. Help files are sought via the following steps:

$ICCONFIGDIR/help/<*command*>.hf
$ICROOT/help/<*command*>.hf
<curdir>/<*command*>.hf

C.3. ICEXEC Control Program

The Interactive COBOL system uses a control program called ICEXEC to coordinate multi-user access to system resources. The following executables **require** the shared area that ICEXEC manages:

icrun	icios	(I/O server started by icnetd)
icruncgi	icrunrs	(ThinClient server started by icnetd)
icsmview		
icwhohas		

All other Interactive COBOL executables can operate with or without ICEXEC. Linux itself does not provide an exclusive open capability and so this is provided by ICEXEC. When ICEXEC is not running, exclusive open is emulated by posting a write-lock on the whole file. A non-exclusive open posts a read-lock on the whole file. Thus, two programs can detect whether a file is opened or open-exclusively by using this mechanism. Care should be exercised when moving from no-ICEXEC to ICEXEC-running as utilities that started in the no-ICEXEC mode will keep running in that mode until they terminate.

D. Conventions

Another aspect of providing a consistent system across multiple operating platforms, is in the command-line interface. The command-line programs use a common command-line syntax across all platforms, and they adhere to the following standard conventions:

1) all switches are composed of a single letter or digit preceded by a hyphen (-);
2) the switches are order independent;
3) the switches ARE case sensitive;
4) lower-case switches imply an action or modification of an action, e.g., `-h' for help;
5) UPPER-CASE switches imply an action with a required argument that must follow with an intervening space, e.g., `-A audit.log' for setting up an auditfile called audit.log.
6) multiple lower-case switches can be combined with one hyphen, e.g., `-axz' for `-a -x -z'.

The following shows how the various conventions for defining syntax will be represented in the Interactive COBOL documentation:

[ ] Brackets enclose optional portions of a format. One of the options contained within the brackets may be explicitly specified or that portion may be omitted.

{ } Braces enclosing a portion of a format means that one of the options contained within the braces must be specified.

| Bar will be used to separate choices when multiple choices are allowed.

... Ellipsis indicates that the previous item can be repeated one or more times.

italic-lower-case Indicates a generic term representing a value that is defined as indicated.

Linux systems support case-sensitive filenames as opposed to Windows that is case-insensitive. All released Interactive COBOL on Linux files are lower-case which is in keeping with most Linux systems. By default, the Interactive COBOL on Linux runtime will convert all COBOL filenames, including program names, to lower-case before looking up that file in Linux. Although Interactive COBOL on Linux can support UPPER-CASE only or mixed-case, we recommend using only one case for filenames to ease portability to case-insensitive environments.

With this in mind, this document will still use upper-case names in the text for specific programs but will always use lower-case in examples and when showing what needs to be entered from the keyboard to run a program.

All examples assume the Bourne shell is being run.

E. Common Switches

E.1. Overall

There are several common switches that appear on all command-line programs except for ICINFO. These are described in detail in the following sections so that the descriptions for each program can be abbreviated. The command-line switch processor scans all the command-line switches, checking for errors. Any errors display an abbreviated startup banner (the program name and revision) to stdout before displaying the error message to stderr and then exiting with a non-zero exit code. If there are no errors to terminate processing prematurely, the common switches are processed. First, if the Help switch is given, an abbreviated startup banner and help text are displayed to stdout after which the program exits normally (i.e. no other switches or arguments are processed). Next, if the Audit switch is given, auditing is enabled. Finally, the Quiet switch, if given, is processed. The program then begins its specific processing by emitting a startup banner, consisting of the program name, revision level, system, and the copyright notice. When it finishes processing, it will emit a trailer message indicating that it is done.

E.2. Audit Switch

The Audit switch will be shown in the syntax as:

-a[:*aflag*] | -A *file*|*dir*[:*aflag*]

Where *aflag* can be a|b|d|p|t|u|da|db|pa|pd|ta|td|ua|ud. *Aflag* modifies the behavior of the switch as:

a	Append	If the current file exists, do Not truncate the file, just append. The Append flag can be used alone or with the Date, Pid, Time, or Username options.
b	Backup	If a previous log file (.lg) exists, rename it to *.lgb and then open a new .lg file. The Backup option can be used alone or with the Date, Pid, Time, or Username options. On Linux, the backup option will break hard links. (Append is not allowed with this option.)
d	Date	Add date in the form of *_YYYYMMDD* before the .lg extension.
p	PID	Add pid in the form of *_NNNN* before the .lg extension.
t	Time	Add dd time in the form of *_YYYYMMDDHHmmsshh* before the .lg extension. (YYYY-year, MM-month, DD-day of the month, HH-hour, mm-minute, ss-second, hh-hundredths of seconds.)
u	Username	Add username in the form *_name* before the .lg extension.

The audit flags (a,b,d,p,t,u) instruct the Audit processing to take a different action then the default for the audit file. The default action is the same as usual, truncate the file to zero on startup.

Note that:

-a	Audit to the default file for this command.
-A *file*	Audit to the specified file.
-A *dir*	Audit to default file in the specified directory.

Audit files contain a copy of any output that was sent to either stdout or stderr, in the same order as it was emitted at execution time (i.e., it may be interspersed). The programs handle this internally, so stdout and stderr can still be redirected. The audit file can be specified to use the default name in the current directory (-a), a user specified name (-A *file*), or the default name in a specified directory (-A *dir*). An audit file is always created if it does not already exist. If it does exist, it is truncated to zero unless the Backup flag is set.

The default audit file name is <command>.lg.

An enhanced auditing facility is available when required. This facility provides verbose messaging and should not be used in a production environment.

The Enhanced Auditing syntax adds the *tflag* options as shown below:

-a[:[*aflag*][:*tflag*][,*tflag*]...]	Audit to <program>.lg
-A *path*[:[*aflag*][:*tflag*][,*tflag*]...]	Audit to file *path*, or *path*/<program>.lg if *path* is a directory

aflag is a|b|d|da|db|p|pa|pb|t|ta|tb|u|ua|ub
a=append, b=backup, d=datestamp, p=process-id, t=timestamp, u=username

tflag is ALL|FILE|ICNETD|IND|MPX|NET|PDF|PGM|REL|SEQ|SP2|SQL|SYSERRS|WEB

ALL enables all sub-system enhanced tracing options.

FILE provides statistics and tracing for lower level file operations (this is very verbose and mostly only useful when requested by Support),
ICNETD provides enhanced tracing for ICNETD,
IND provides Indexed i/o statistics and open/close tracing,
MPX provides enhanced tracing when using network multiplex i/o from the runtime, iclogs, icrunrc, icodbcdr (any program that uses the network),
NET provides for enhanced network tracing for any network traffic,
PDF provides enhanced tracing when using the PDF facility from the runtime,
PGM provides CALL/EXIT/CANCEL tracing, (including builtins),
REL provides relative i/o statistics and open/close tracing,
SEQ provides sequential i/o statistics and open/close tracing,
SP2 provides enhanced tracing when using SP2 or QPR from the runtime or icrunrc,
SQL provides enhanced tracing when using ISQL from the runtime,
SYSERRS provides enhanced tracing for System errors,
WEB provides enhanced tracing when using any internet type facility like email from the runtime.

These trace flags for enhanced auditing should be used when requested by support or to track down problems as the logs can be very verbose.

E.3. Quiet Switch

The Quiet switch will be shown in the syntax as:

-q

The Quiet switch works by suppressing all output that is emitted to stdout. The most obvious effect is that it suppresses the usual banner and trailer messages that are emitted to stdout as the program starts and terminates. Because it is suppressing stdout, the Quiet switch may also suppress other parts of the usual output.

E.4. Help Switch

The Help switch will be shown in the syntax as:

-h|-?

The Help switch displays a summary of the command-line syntax, the switches and what they do, and the applicable environment variables.

F. Filename Extensions

Interactive COBOL requires that the extension for certain specific types of files to match those given in the following table except for those marked *defacto*. Those marked *defacto* are only the most common extensions used for these purposes and are not required. All Interactive COBOL release files will conform to these *defacto* standards.

Those extensions marked as this sentence is marked are extensions in some older revision of Interactive COBOL or ICHOST but are handled in some special cases by current Interactive COBOL utilities.

Common extensions used by Interactive COBOL include:

.cd	old ICHOST COBOL program file
.cf	old Configuration file (pre 3.30)
.cfi	Configuration file (.ini)
.cl	Library file
.co	COBOL Source program (card format) *(defacto)*
.cob	AOS/VS COBOL text format source
.cx	COBOL Program file
.dt,.nt	Pair of files, ICPACK data and index temporary files
.er	Error file *(defacto)*
.fa	File attribute file
.fp	Failsafe protection file
.gsy	Global symbol file for the IDE
.hf	Interactive COBOL help files
.icp	Project file for the IDE
.lg	Audit / Log file *(defacto)*
.lgb	backup Audit / Log file (*defacto*)
.lk	Link file
.ls	List file *(defacto)*
.ms	Message file
.pd,.dd	Pair of files, older revision COBOL program file (program and data)
.pq	Printer control file
.pt	old Printer translation file (pre 3.30)
.pti	Printer translation file (.ini)
.rp	old Remote protection file (MS-DOS only)
.sd	ICRUN Sort data file (temporary)
.sr	COBOL Source program (text format) *(defacto)*
.st	ICRUN Sort tag file (temporary)
.sy	COBOL Symbol table file
.td	old Terminal description file (pre 3.30)
.tdi	Terminal description file (.ini)
.tmp	Temporary file *(defacto)*
.udb	U/FOS database
.xco	COBOL Source program (extended card format) (new in 5.40)
.xd,.nx	Pair of files, COBOL ICISAM file (data and index portion)
.xdb	ODBC database definition file (.ini)
.xdt	ODBC table definition file (.ini)
.xl	Log file (pre 5.40)
.xlg	Generation log file (pre 4.00)

All Interactive COBOL utilities support mixed-case filenames. If a utility needs to add a filename extension, e.g., .xd or .nx, etc., it searches back from the end of the simple filename for the first alphabetic character. If it finds an upper-case alphabetic it will use an upper-case extension, otherwise a lower-case extension is used. For example "iccheck DATAbase1" and "iccheck 12345" would use the lower-case extensions `.xd' and `.nx' for the ICISAM file, while "iccheck dataBASE52" would use the upper-case extensions `.XD' and `.NX'.

G. Exit Codes

All command-line programs return exit codes that provide an indication of the success or failure of the program. These are returned through the exit code. In general, the following codes will be returned:

0	The program completed without errors.
1	The program ran, but some items it processed had errors. For example, ICCHECK checked a series of files, and some of them were corrupt.
2	The program was running, but was terminated by an operator interrupt or external abort.
3	The program was running, but was terminated by some fatal internal error. For example, the compiler was running but detected that its virtual memory manager had run out of memory unexpectedly.
4	There were command-line errors so the program did not perform any of the requested function(s).
5	The user was not authorized to execute the program or perform a requested operation, so the program did not run.
6	The program experienced an error during its initialization phrase and could not execute. For example, it could not allocate sufficient memory to perform its function.
7	Help was requested
8-9	Reserved for future `common' errors.
10-	These codes are specific to each program and will be documented with each program.

All of the programs support exit codes 0 through 9 with the meaning described above.

H. Common Environment Variables

H.1. Overall

There are several common environment entries that most command-line programs use. These are described in detail in the following sections so as to not be duplicated under all program descriptions. Other environment variables that are more program specific will be described under each program.

H.2. ICROOT

ICROOT specifies the Interactive COBOL root directory. ICROOT is used to find directories like help and other important files needed to operate the **ICOBOL** runtime environment. ICROOT should always be set and it should always be set to the current installation directory on a production system.

The syntax is:

```
ICROOT=dir
```

Where

dir

Specifies the main install directory. Usually this should be set to the current revision directory.

If ICROOT is not set, the current directory is used but the system may not function correctly.

H.3. ICCONFIGDIR (Added in 4.70)

ICCONFIGDIR specifies a directory for customized system files. ICCONFIGDIR provides a mechanism for user-customized versions of files that come as part of the standard Interactive COBOL installation. It should specify a directory that is separate from the installation directory so that when the Interactive COBOL installation is upgraded, the customized files will not be affected.. All programs will use ICCONFIGDIR before they use ICROOT when searching for system files, so, for example, it could be used to store terminal description files (.tdi) that have a custom color configuration, or non-English versions of the help files.

The syntax is:

```
ICCONFIGDIR=dir
```

Where

dir

Specifies the directory where to find customized help, messages, print, or term files.

If ICCONFIGDIR is not set, then ICROOT is used. This would be the same as version before 4.70.

H.4. ICTMPDIR

ICTMPDIR specifies a directory to which programs may write any temporary files.

The syntax is:

```
ICTMPDIR=dir
```

Where

dir

Specifies a valid pathname for the directory in which any needed temporary files are to be written.

If ICTMPDIR is not set, the current directory is used.

Some of the programs that look for the ICTMPDIR environment variable are ICOBOL, ICRUN, and ICSORT.

H.5. ICPERMIT_MACHINE

ICPERMIT_MACHINE specifies the default license server machine and TCP port.

The syntax is:

```
ICPERMIT_MACHINE=machine[:port]
```

or

```
ICPERMIT_MACHINE=:port
```

Where

Machine

Specifies an ip-address or a machine-name on which an ICPERMIT license server is running. The default is the current machine (localhost).

Port

Specifies the TCP port which ICPERMIT is running. The default is 7334. The license server and the clients MUST use the same port.

Some of the programs that look for the ICPERMIT_MACHINE environment variable are ICOBOL, ICRUN, the ICNETD servers (ICIOS, ICRUNRS, ICLOGS, ICSQLS), ICPERMIT, and any other **ICOBOL** licensed executables.

H.6. Executable Name

All command-line utilities support an environment variable of the same name as the utility in upper-case. For example, 'iccheck' will recognize the variable ICCHECK. The environment variable can contain command line options for the utility which will be processed prior to any options actually present on the command line. If such an environment variable is present, the utility will display the complete set of options at startup.

I. Reporting Problems

If you have problems, please first review the manual to recheck your installation and operation. Next, check APPENDIX C for a list of general problems to see if any discuss your particular incident. Next, review the moments just before the problem occurred to see if you changed anything, i.e., used ICCONFIG, added a new device driver, changed attributes, etc.

If your problems still persist, email us at support@icobol.com with a description of the problem. Please include the information listed below:

1) Describe only one problem at a time

2) Send a copy of the output of ICINFO from the system with the problem. The latest revision of ICINFO can be downloaded from the web site (www.icobol.com). If the system in question is an older version, make sure to inform us which version of ICOBOL has the problem.

3) If the problem is a COBOL problem, a simple test program that reproduces it should be provided. The shorter the better. Include the source and .CX files for the program, along with any data files needed. If this program requires user input (name, password, data-entry, etc.), please provide a script for how to logon and duplicate the problem.

4) If data files are being corrupted then please send `before' and `after' images of the file.

5) If the problem is with a utility, send the exact command line of how the utility was invoked.

6) For problems involving devices such as terminals, printers, etc., please send a copy of the current configuration file.

7) Media should be in compressed tar (.tgz) format. If it is too large to attach to an email, you can request a link from us to be able to upload the file to us.

Other items that would be helpful to include are:

Is the problem repeatable?

Can you work around the problem?
If so, how?

What is the easiest way to get the problem?

What is the importance of obtaining a solution to the problem?
(Urgent, moderate, low, nuisance)

II. INSTALLATION

A. Introduction

This chapter provides for the initial installation of the Interactive COBOL on Linux product including required Linux kernel changes along with concerning running Interactive COBOL on Linux on various platforms. Interactive COBOL on Linux (<os>XXX.tgz) includes all parts (runtime, development, ICODBC, ICNETD) of the Interactive COBOL release. The readinst.txt file should also be examined for updated installation guidelines. The XXX represents the current revision. I.E. 500 for 5.00.

Flavors of Interactive COBOL on Linux should generally be used only on the indicated operating system. See the readme file for specific revision levels.

Interactive COBOL on Linux does not provide any device drivers for terminal connection boards. These device drivers generally come with the terminal connection boards(s) and are specific for each Linux system.

After the installation you will need to use the Interactive COBOL ICCONFIG utility to build custom configuration file(s) and any custom terminal description files (.TDI) that are specific to your particular installation.

B. Software Installation

B.1. Download the software

This section provides the necessary steps to get the Interactive COBOL on Linux software loaded.

The installation packages for Interactive COBOL on Linux are available for download via a browser from our website at http://www.icobol.com/download/current.php. The installation package that you choose will depend on the version of the operating system you are using. Our standard is Red Hat Enterprise Linux (RHEL) or its free community-built version, CentOS (www.centos.org). We currently have downloads for 3 versions of RHEL - versions 6, 7, and 8. Since version 6 was available in both 32-bit and 64-bit, there are 2 corresponding downloads for version 6. Versions 7 and 8 are only available in 64-bit.

The installation files are packaged as a compressed tar file and are named with an operating system prefix followed by a version number, followed by a package suffix. The OS specifier uses 'ln' to indicate it is for Linux, then ‘6', ‘7', or ‘8' for the OS version, 'x32' or 'x64' to according to whether the Linux installation is 32-bit or 64-bit, followed by a dot and then a 3 digit number that indicates the revision. The full installation package has no suffix. The 'info' suffix indicates that the package contains just the ICINFO utility. The 'odbc' suffix indicates that the package contains just the ODBC driver. The 'tc' suffix indicates that the package contains just the thin-client software. So, for a version 5.40 full installation for a RHEL 8 64-bit installation, the downloaded file will be ln8x64.500.tgz.

In the documentation that follows, we will refer to the 3 OS versions by the short-hand ln6, ln7, or ln8 prefix that matches the installation file.

B.2. Loading the software

The software can now be loaded using the *tar* command with the appropriate specifier for your Interactive COBOL media. This load should be done in the directory in which you wish the Interactive COBOL directory (icobol.<rev>) to be created, usually /opt, which is the location assumed by the documentation and the scripts. The Interactive COBOL directory will be loaded in the form, "icobol.<rev>", where <rev> will be the current revision of the software. We will use icobol.XXX to refer to this directory. **This load should be done as super user.** With this naming convention multiple revisions of **ICOBOL** can be installed at the same time and the appropriate paths must be used to access the needed revision. See the *readic.txt* file for what revisions can be mixed.

```
su
umask 0          [suggested mask]
cd /opt          [or the directory where you wish the icobol.XXX directory to be created]
tar -xvf path-to-<os>XXX.tgz
```

While performing the above steps, you should see the main Interactive COBOL directory named icobol.<revision> along with its subdirectories and their files loaded. These directories contain the required files for Interactive COBOL on Linux.

You should now save your release media (or downloaded file) for future reference.

The *readme* files (*readic.txt* is the main one) are in the icobol.XXX/docs directory they should be read for any new information that has not been put into the Interactive COBOL manual set.

```
cd icobol.XXX/docs
pg read*.txt
```

The icobol.XXX directory holds all the executable files needed for Interactive COBOL operation for the particular base operating system. For 64-bit operating systems there is an additional subdirectory named 'x86' that holds 32-bit executables for the runtime and thin client. There is a further sub-directory to that named 'icnet' that has the 32-bit surrogates for ICNETD. The x86 subdirectory can be added to the user's PATH if using a 32-bit runtime is necessary..

The *icobol.XXX/examples* subdirectory offers several directories *cgi*, *config*, *icodbc*, *print*, *programs*, *scripts*, *terminfo* which additional files for those areas.

A sample COBOL program called *logon* has been included in the *examples/programs* directory along with its source file that shows examples of some Interactive COBOL-specific and other non-specific features. Some of the features included are: use of terminal Print Pass Through, disk space, directory listings, and the Get Error Message ability. The logon program source file is *logon.sr*.

The *icobol.XXX/install* subdirectory provides installation scripts.

The file *icobolrc* in the *install* subdirectory is a partial shell script that can be added to the default *.profile* script to allow the required Interactive COBOL environment entries to be setup when a user logs in to Linux.

B.3. Installing the software

In most environments shared objects are used to access common code. These shared objects are icrun32.so.<*rev*>, icrun64.so.<*rev*> used by runtime executables, and icsys32.so.<*rev*> and icsys64.<*rev*> for almost all of the remaining**ICOBOL** executables. When used, these objects rev check themselves for integrity.

Additional shared objects include icodbc32.so.<*rev*> and icodbc64.so.<*rev*>. These are documented in their own readme file named readodbc.txt

There was a major change in how service daemons are managed from ln6 (and before) versus ln7 (and later). In ln6 and before, daemons were managed by scripts in /etc/init.d and through the utilities chkconfig and service and those daemons ran under a master process called init. That model was based on System V UNIX systems that were prevalent when Linux systems were first developed. Beginning with ln7 the system was changed to use a master process called systemd and a utility called systemctl. The systemd process manages some of the operations that were common to to init.d scripts and older daemons. This results in a simpler design for the daemons, their scripts, and their configuration. As a result, the *installic* script that is in the ln6 package is like the pre-5.40 script and uses the /etc/init.d, chkconfig, and service methods; whereas, the *installic* script uses the systemd and systemctl methods. This caused some changes in how secondary scripts, such as *icobolrc* are used.

Installation modes:

METHOD ONE (Recommended for production systems) Uses install/*installic* script.

For ln6 Installs:

This is a full install and sets up the needed links (/opt/icobol) for the actual release directory that was loaded above, /var/log/icobol for log files, /etc/opt/icobol for configuration information, structure for the services in /etc/init.d/icobol (for ln6), and /var/opt/icobol for system data like the failsafe, system.pq, and tmp, and the links for the shared objects in the appropriate /usr/lib and /usr/lib64 directories.

As superuser, first run the install/*installic* script. This will set up the needed structure as specified above.

Second edit the configuration file /etc/opt/icobol/icobolrc. This file has some default environment entries that will be used if not overridden. These include ICROOT, ICCONFIGDIR, ICCODEPATH, ICPERMIT_MACHINE, ... just to name a few of the more important environment entries. This file is used as the base configuration file for services in ln6 and it can also be used in the user's .profile to set up the base environment.

Next review and edit the service startup configuration file /etc/opt/icobol/icobol.conf. The defaults in this file assume that the license file is system.lic is in the ICCONFIGDIR directory along with the system configuration, system.cfi, file. If ICNETD needs to be started this should be enabled in this script. Once icobol.conf has been updated the Linux *service* command can be used to start and stop services. Some examples are:

```
# service icobol start
# service icobol stop
```

See the file install/icobol for more information.

For ln7 or ln8 Installs:

This is a full install and sets up the needed links (/opt/icobol) for the actual release directory that was loaded above, /var/log/icobol for log files, /etc/opt/icobol for configuration information, and /var/opt/icobol for system data like the failsafe, system.pq, and tmp, and the links for the shared objects in the appropriate /usr/lib and /usr/lib64 directories.

As superuser, first run the install/*installic* script. This will set up the needed structure as specified above.

Second edit the common configuration file /etc/opt/icobol/icobolrc. This file has some default environment entries that will be used if not overridden. These include ICROOT, ICCONFIGDIR, ICCODEPATH, ICPERMIT_MACHINE, ... just to name a few of the more important environment entries. We recommend using this file in the user's .profile to set up the base operating environment for **ICOBOL**.

Next review and edit the service startup configuration files in /etc/opt/icobol. There is a master file, named services.conf, that configures which services you are using that need to be started and stopped. Additionally, there is a configuration file for each of the three service daemons: ICPERMIT, ICEXEC, and ICNETD named icpermit.conf, icexec.conf, and icnetd.conf, respectively. In ln6 and pre-5.40, this information was combined in one file - icobol.conf.

If you are upgrading a pre-5.40 installation on an ln7 or ln8 system, use your previous icobol.conf file and /etc/init.d/icobol file to help guide your editing. If this is a new install, these files may require some minor updates. If this is an upgrade install of a 5.40 or later system, these files usually do not need to be changed.

The icpermit.conf has default values for the license file (system.lic), the failsafe file, and logging. Similarly, icexec.conf are has default values for the **ICOBOL** system configuration (system.cfi), the printer control file (system.pq), and logging options. The icnetd.conf file has some basic default values, but typically it requires additional customization depending on which surrogates (icios, icrunrs, icsqls) are being used. Once the changes are complete, the Linux *systemctl* command can be used to start and stop individual services, or the supplied scripts *icstart*, *icstop*, and *icstatus* can be used to operate on all of the enabled service as a group. Some examples are:

```
# systemctl start icpermit.service (the .service suffix is optional)
```

```
# systemctl stop icnetd
# systemctl status icexec
# icstart
# icstop
```

For All Installs:

The *installic* script should be run any time a new **ICOBOL** revision is installed and unless requested by the readme.txt file the scripts that had been edited in /etc/opt/icobol do not need to be updated.

The *uninstallic* script can be run to remove **ICOBOL** from the system.

Some historical scripts are included *installso* and *removeso* that can be used to tailor an install process.

The *installso* script should be run from the install directory as super-user. By default it will install shared objects into the appropriate /usr/lib64 and /usr/lib directories as symbolic links.

If this method is used, then the *installso* script will need to be run each time a new revision of **ICOBOL** is installed. The *installso* script is provided in the install sub-directory.

METHOD TWO (Only recommended for testing and debugging)

Set the system environment variable that is used to find shared objects to include the new ICROOT directory where the shared objects are to be found. In most cases this is the LD_LIBRARY_PATH environment variable. Thus to add the new ICROOT to the LD_LIBRARY_PATH you could do:

```
if [ "$LD_LIBRARY_PATH" = "" ]; then
   LD_LIBRARY_PATH=$ICROOT
   export LD_LIBRARY_PATH
else
   LD_LIBRARY_PATH=$ICROOT:$LD_LIBRARY_PATH
fi
```

This could be done in the script that sets up the PATH and ICROOT for your **ICOBOL** executables. If this was a separate script before then just add the above lines. If this was part of the users' startup in .profile then just add it there.

This method should be used when you do not want to change the /usr/lib directory and/or you will be running several versions of **ICOBOL** for development and/or testing purposes.

B.4. Initial Test

At this point an initial test would be to run a particular utility to ensure the install was successful.

Try an iccheck for example.

ERRORS:

If you get an error like:

```
  error while loading shared libraries: icsysnn.XXX.so: cannot open shared object
or
  dynamic linker: icstat: icsysnn.XXX.so is NEEDED but object does not exist.
```

when running an **ICOBOL** executable, then the system cannot find the indicated shared object. Either METHOD ONE or TWO must be done to allow the file(s) to be found.

In general, the shared object icsys<nn>.so.<rev> should be available for most **ICOBOL** executables, and the icrun<nn>.so.<rev> shared object should be available for all runtime executables.

III. LICENSING (ICPERMIT)

A. Introduction

This chapter discusses the license description file and how to use ICPERMIT, the global license server daemon that provides authorization information to any Interactive COBOL processes that requires licensing.

Interactive COBOL on Linux requires a valid license to be present for various programs to be executed. ICPERMIT is the authorizing program that reads a license description file (default system.lic) and authorizes the specified product(s) for the given user count. On Linux several different protection methods are used to authorize a license. These include a serial protection device, with a matching serial number to the license or a MAC address which uses the Media Access Control address of a network (usually ethernet) interface.

ICPERMIT allows licenses to be shared over a TCP/IP-based network. Licenses can be shared among machines of different types (Windows and Linux).

If remote licensing is being used, ICPERMIT should not be running on the local machine but must be available over a network from a central server.

B. License Description File

The license description file provides the unique information needed to license an individual site for the various combinations of product(s) and users. In most cases it must be used with a unique protection device either parallel or USB (Windows) or serial (Linux). Additionally, on Linux and Windows the MAC address of an ethernet card can be used.

The license description file is provided electronically as an email attachment. The file is a simple text file that should be copied to your Linux system. We recommend that you copy the file to /var/opt/ icobol/system.lic.

PRODUCT ACTIVATION KEY

The PRODUCT ACTIVATION KEY contains the license and key information to authorize licensed products. Each KEY shows one or more licensed products in the following fashion:

```
* For SERIAL DEVICE SN:  00000766
* ICOBOL Runtime License
LIC 01-SERIAL-00000766-ICRUN-50-A-ANYWHERE-00033
KEY 2fgy87k1ms-8u1op1mn98-jhtrewsa-j8h6frd5s4-cxwzbnmk87-98up0kmngf
```

Where

The first line is a comment showing the type (MAC, PARALLEL, SERIAL, USB, etc.) of license with its serial number(SN:). (Linux systems support MAC and SERIAL types.)

The second line is a comment showing the licensed product.

The third line (starting with LIC) is the text description of the license showing the format revision (01), license type (SERIAL), serial number (00000766), licensed product (ICRUN), revision (50), options (A), operating system (ANYWHERE), and user count (00033).

The final line (starting with KEY) is an encrypted version of the license that can spill across multiple lines as needed.

NOTE: The letters "i", "l", "q", and "v" are never used to prevent confusion with the number 1 or the letters "g" and "u".

Lines in the license description file that start with a `*' are comment entries only. Comment entries and blank lines are ignored.

If the license type is PARALLEL, SERIAL, or USB then this license must be paired up with the parallel, serial, or usb protection device with the same serial number to allow authorization. If the license type is MAC, then this license will only run on a particular hardware configuration with its unique MAC network address as provided by the network interface card (NIC).

ICPERMIT with the Check switch (-c) can be run to check the file and protection method.

Licensed products for each type of license currently include:

All runtimes (ICRUN/ICRUNW, ICRUNCGI, ICRUNRS, etc.) use **ICOBOL** Runtime Licenses,
Runtimes that use SP2 and/or FormPrint use **ICOBOL** SP2 Runtime Licenses,
All requests to ICNETD (client/server) use **ICOBOL** Network Server Licenses,
this includes the ICIOS, ICRUNRS, and ICSQLS surrogates,
ICOBOL and ICIDE use **ICOBOL** Development Licenses,
ICSP2 uses **ICOBOL** SP2 Development Licenses,
ICQPRW uses **ICOBOL** FormPrint Development Licenses,
User-Programs built using the User Library use **ICOBOL** Application Interface Licenses,
(Licensed options on Linux include IND_API)
ODBC-enabled programs using the **ICOBOL** ODBC Driver use **ICOBOL** ODBC Driver Licenses.

C. Serial Protection

Serial protection devices are supported for Interactive COBOL on Linux.

If your license description file specifies "For SERIAL DEVICE SN: xxxxxxxx" then that serial protection device (SN: xxxxxxxx) must be installed on an asynchronous RS-232C DTE serial line that is configured at 9600 baud, 8 bits, no parity, and 1 stop bit. Pins 4 (RTS) and 20 (DTR) are required to supply the serial protection device with positive voltage; both pins are required for the serial protection device to work. These pins must be asserted (held positive with respect to line 7) at all times. The serial protection device causes no interference to an attached device or character transmitted to that device other than the specifications must match above. The serial protection device provides a unique identification value to which an individual license description file is keyed. The serial protection device is passive to all character traffic unless it sees a particular 16-byte sequence. In normal use, this "wake-up" sequence should not present a problem for other devices attached to the line.

If your computer does not provide an asynchronous port with an RS-232 25-pin connector, then you must provide an adapter that does. The following discussions assumes an RS-232 25-pin connector.

If the Computer Port provides a DTE connection with power on pins 4 (RTS) and 20 (DCD) and is a male connector then insert the Serial protection device as shown in FIGURE 1 making sure that the female end of the serial protection device points toward the computer.

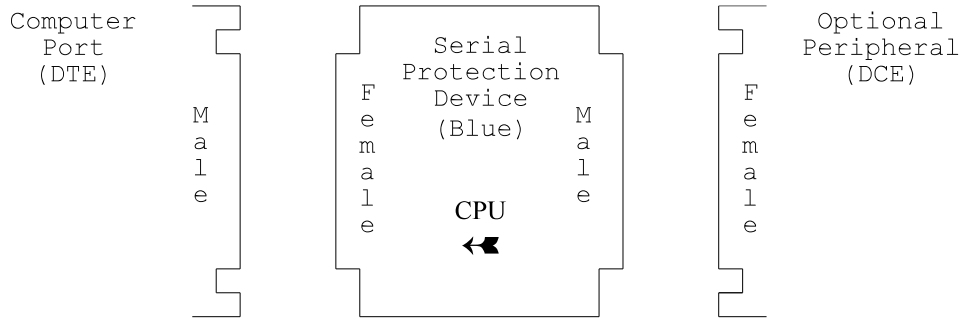


FIGURE 1. Standard Serial Protection connection

If the computer port is a female connector, then install a male-male gender changer before inserting the Serial protection device, as shown in FIGURE 2 making sure that the female end of the serial protection device points toward the computer. You may wish to install a female-female gender changer on the other side of the serial protection device to provide the same interface as when you started.

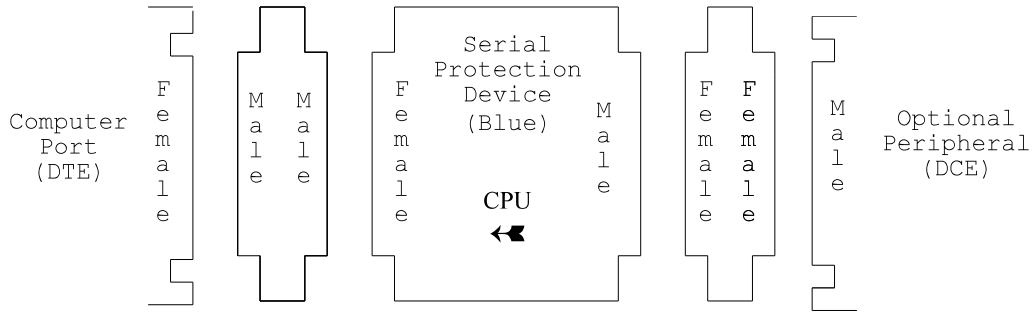
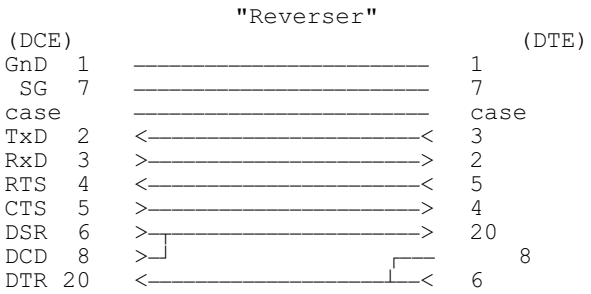


FIGURE 2. Standard Serial Protection connection (with gender change)

If the Computer Port provides a DCE connection with power on pins 5 (CTS), 6 (DSR), and 8 (DCD) then a "Reverser" can be inserted to change the DCE connection into a DTE connection. This "Reverser" should have the following wiring diagram.



Using "Reverser" adapters, insert the Serial protection device as shown in FIGURE 3, along with any gender changers necessary to make sure the female connector of the Serial protection device points toward the computer.

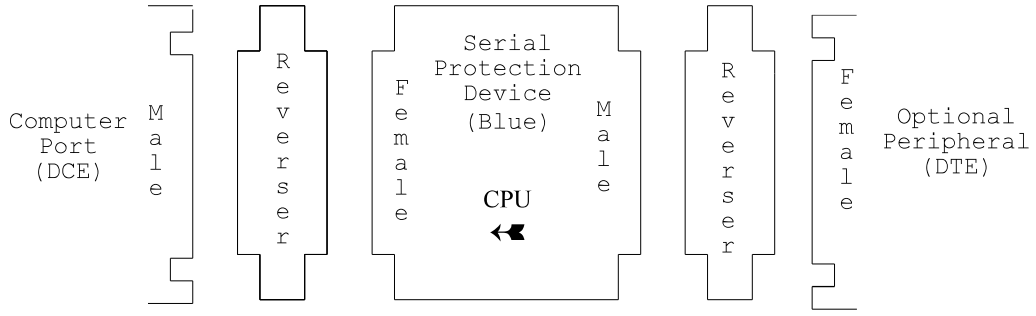


FIGURE 3. Non-Standard Serial Protection connection

If the computer port provides a DCE connection with power on pins 4 (RTS) and 20 (DTR) then a modified-Reverser can be built to only flip pins 2 and 3. Use the above as guidelines.

If the computer port does not provide power on the needed pins then move the serial protection device to a different asynchronous port of the computer -- a modem port is ideal for this purpose.

Notes: 1. A DTE connection transmits on pin 2 (receives on pin 3) and signals its willingness to communicate on pins 4 (RTS) and 20 (DTR).

2. A DCE connection transmits on pin 3 (receives on pin 2) and signals its willingness to communicate on pins 5 (CTS), 6 (DSR), and 8 (DCD).

3. An RS-232 Tester (or Diagnostic Indicator) can be used to determine the type of connection a port provides. RS-232 Testers generally have Green and/or Red LED's/LCD's to indicate power. If pin 2 (TxD) has power then the port provides a DTE connection. If pin 3 (RxD) has power then the port provides a DCE connection. For the serial protection device to work on a DTE port, pins 4 (RTS) and 20 (DTR) must have power. For the serial protection device to work on a DCE port using a "Reverser" configuration, pins 5 (CTS), 6 (DSR), and/or 8 (DCD) must have power.

D. MAC Protection

If your license description file specifies "For MAC Device xx-xx-xx-xx-xx-xx", then a network card in the machine must have a MAC address with the given value. The icinfo utility can be used to display the current MAC address(es) available on a machine. The command would be: icinfo -n.

E. ICPERMIT

E.1. Syntax

The syntax for ICPERMIT is:

```
icpermit [-a[:aflag]|-A file|dir[:aflag]] [-c] [-f] [-F faildir] [-h|-?]
   [-i] [-L licedesc] [-M machine[:port]] [-N w] [-O a|c|d|e|i|m|p|t|u|v]
   [-P device] [-q] [-s] [-t] [u] [-V hlpPrtu]
```

Where

-a[:*aflag*]|-A *file*|*dir*[:*aflag*] (Audit)
Enables auditing (default icpermit.lg). Where *aflag* is a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-c (check)
Check a license description file for validity and query any specified device, do not install. (Check can be used even when another ICPERMIT is running.)

-f (Failsafe)
Use an existing failsafe security file instead of the defined protection device.

-F *faildir* (Failsafe location)
Specifies where to locate the failsafe security file. The failsafe security file, system.fp, is located in this directory if given, otherwise the current directory is used.

-h|-? (Help)
Display help text.

-i (Info)
Put out Info messages.

-L *licedesc* (License file)
Specifies the license description file. The default is system.lic.

-M *machine*[:*port*]] (Machine)
Specifies the license server machine and optional TCP port. The default is the current machine (localhost) and port 7334. *Machine* can be an ip-address or a machine-name.

-N w (No)
Specifies a No option: Valid options are: w=No Warnings.

-O a|c|d|e|i|m|p|t|u|v (Operation)
Specifies an operation to perform on an already running ICPERMIT. Valid operations are:

a (Amplify)	Amplify (turn on) tracing
c (Check)	Check if running,
d (Disable)	Disable any new connections,
e (Enable)	Re-enable new connections,

i (Info)	Display license information,
m (Mute)	Mute (turn off) tracing,
p (Post)	Cause the current connection/license information to be written to the log file,
t (Terminate)	Terminate,
u (Update)	reread the license file and re-authorize licenses
v (verbose)	Display of license information according to the -V options If no -V is specified, the default is -V lhup.

-P *device* (serial Protection device location)

Specifies the serial device on which the serial protection device is located. If not specified, the current console, /dev/tty, is used. This should always be specified in ln7 and ln8.

-q (Quiet)

Enable quiet operation.

-s (systemd)

Run the daemon in systemd mode (set in ln7 and ln8)

-t (Tracing)

Start with tracing mode amplified (turned-on).

-V hlpPrtu (-O v format)

The order of the letters determines the order of the output columns and the sort order.

h (host)	Host node name
l (license)	License type name
p (program)	Program name
P (Pid)	Process ID of client
r (revision)	Revision number of client
t (time)	Time client connected
u (username)	User name of client

Note: The Check Operation (-O c), the Check license (-c), and the Post Operation (-O p) do not require super user.

ICPERMIT uses TCP/IP to communicate with processes needing authorization in the default case. If for some reason TCP is not installed on the machine OR no network license connection should be allowed then the :port selection can be set to zero as :0. Only local sockets will be used in this case. ICPERMIT_MACHINE must be set for any executable that uses licensing.

ICPERMIT looks for the environment variables ICPERMIT and ICPERMIT_MACHINE.

E.2. Description

ICPERMIT can run in one of several modes:

Check mode	(-c)	validates a license description file and device
License mode	(neither -c or -O)	provides license authorization
Operation mode	(-O)	performs an ICPERMIT operation

Check mode (-c)

Check mode instructs ICPERMIT to validate the given license description file and then query the protection device to confirm the license. ICPERMIT in check mode does not interfere with an already running ICPERMIT or whether local or remote licensing is in effect. Check mode does not require super user.

After entering a license description file, ICPERMIT should be used to check that file.

Below is the output from a successful check.

```
[root@ln7x64vm ~]# icpermit -O c
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
icpermit is running on pid 11767 of computer localhost, using port 7334
icpermit is finished
[root@ln7x64vm ~]#
```

SCREEN 1. ICPERMIT CHECK

License mode

License mode instructs ICPERMIT to authorize processes that require Interactive COBOL licenses. ICPERMIT uses the information in the license description file to provide any required authorizations. License mode requires super user.

Starting and stopping ICPERMIT does not impact ICEXEC or the ICNETD server.

For a production system, ICPERMIT should be started when the system moves from single user or maintenance mode to multi-user mode by whatever method is appropriate. The standard install script will configure the needed services so the service command can be used.

If ICPERMIT is started at startup and a serial protection device is being used, make sure that the particular serial device driver on which the serial protection device is located has been initialized by the operating system. This is especially important for serial lines that reside on cluster controllers or otherwise smart serial boards in which code must be downloaded and executed on the serial board/controllers before the lines are active.

If ICPERMIT is started on a pseudo-tty or on a tty that may log off from Linux such that ICPERMIT can no longer log to the starting console, then the Quiet switch (-q) should be given when starting ICPERMIT to prevent it from logging to the default console and possible hanging on a log write.

When ICPERMIT starts it takes the following steps:

1) Processes and validates any command line switches.
2) The audit file is opened. If the audit file does not exist, it is created.
3) Checks that the user is in super user mode.
4) Opens a TCP/IP socket on the specified port to communicate with licensed products.
5) Reads the license description file and provides authorization based on that information.
    a) If the Failsafe switch (-f) was given the failsafe security file is used to authorize users until a new protection device can be installed. This authorization is only temporary and will expire after the indicated period.
    b) For a serial protection device, ICPERMIT interrogates the device as specified by the license to validate the license provided and authorize users. For a MAC address the appropriate NIC interface is interrogated.
    c) If authorized by step b above, a new failsafe security file (system.fp) is written to the appropriate location for subsequent use with the Failsafe switch (-f) if the protection device fails for some reason.
6) On ln6 systems ICPERMIT then detaches from the current console and becomes a daemon. On ln7 and ln8 systems under systemd, this step is handled by systemd.
7) Waits to authorize users.

If at any point in the above steps ICPERMIT detects an error, it displays the appropriate message and terminates. On ln6 systems, if step 6 succeeds, the user on the invoking console will be back at the shell prompt with an exit code of 0 and at least two (2) messages from the ICPERMIT daemon that will show up as:

```
(date) (time) icpermit (pid#):Info: Detached from login session context.
(date) (time) icpermit (pid#):Info: The license server is ready
```

If the license description file itself has a problem, a message like the following will be given showing the invalid line:

```
[root@ln7x64vm ~]# icpermit
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing license description file: /usr/cobol/system.lic
Error:  Invalid or mis-matched license information: Line 6
Error:  No licenses are authorized
icpermit is finished
[root@ln7x64vm ~]#
```

SCREEN 2. ICPERMIT STARTUP WITH LICENSE DESCRIPTION ERROR

If an error is generated in step 5 using the serial protection device then one of the following may be true: the serial protection device is attached to the wrong port or not attached, or wiring to the protection device is incorrect. SCREEN 3 shows the case of trying to talk with the serial protection device on the current console and no device is installed. As can be seen, after the "Processing protection device authorization" line, a line starting with "EGAN SYSTEMS INC" is sent to the serial protection device to interrogate it and since the device was not present to intercept the sequence it came to the screen. Either attach the serial device to this line or start ICPERMIT using a different serial line (-P *device*).

```
[root@ln7x64vm ~]# icpermit
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing license description file: /usr/cobol/system.lic
Info:   Processing protection device authorization, please wait..
EGAN SYSTEMS INCpoernvd83mnds8vs,mnv7gf,3476fvm237ak43m23fa7fmn4%
Error:  Device timeout: /dev/tty
Error:  No licenses are authorized
icpermit is finished
[root@ln7x64vm ~]#
```

SCREEN 3. ICPERMIT SERIAL STARTUP WITH ERROR

SCREEN 4 shows the case of trying to talk with the serial protection device on another serial line and the serial protection device does not respond in a reasonable amount of time. In this case the serial protection device is probably not installed, not installed properly, or another process is on the line (i.e. multiple readers).

```
[root@ln7x64vm ~]# icpermit -P /dev/ttyS0
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing license description file: /usr/cobol/system.lic
Info:   Processing protection device authorization, please wait.
Error:  Device timeout: /dev/ttyS0
Error:  No licenses are authorized
icpermit is finished
[root@ln7x64vm ~]#
```

SCREEN 4. ICPERMIT SERIAL STARTUP WITH ERROR

In the above case if the Error was "The file was not found", then the particular serial line could not be found to be opened.

If an incorrect serial device is found, the following will be generated:

```
[root@ln7x64vm ~]# icpermit -P /dev/ttyS0
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing license description file: /usr/cobol/system.lic
Info:   Processing protection device authorization, please wait..
Error: License description file does not match protection device
Error:  No licenses are authorized
icpermit is finished
[root@ln7x64vm ~]#
```

SCREEN 5. ICPERMIT SERIAL STARTUP WITH ERROR

Once ICPERMIT becomes a daemon, it sends any messages to the invoking console (unless the Quiet switch was given) and the audit file in the following format:

(*date*) (*time*) icpermit (*pid#*):*severity*:*text*

Where

date is the current date,
time is the current time,
pid# will be the pid on which ICPERMIT is running and is the pid# that must be used on a KILL to terminate ICPERMIT,
severity shows the level of message severity, consisting of PANIC, ERROR, INFO, and WARNING, and
text message contains specific information.

The audit file (icpermit.lg) is created or erased every time ICPERMIT is invoked.

If ICPERMIT detects any error conditions while it is running, an appropriate message is sent to the audit file and the invoking console.

SCREEN 6 shows an example of the output when ICPERMIT is invoked from a terminal. The first two "Processing " messages come from the initial icpermit process. The final two messages (the ones starting with icpermit (pid#):) come from the daemonized icpermit process.

```
[root@ln7x64vm ~]# icpermit -P /dev/ttyS0
icpermit Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing license description file: /usr/cobol/system.lic
Info:   Processing protection device authorization, please wait..
 (date)(time)icpermit (263):Info: Detached from login session con
 (date)(time)icpermit (263):Info: The license server is ready
[root@ln7x64vm ~]#
```

SCREEN 6. SUCCESSFUL ICPERMIT STARTUP

If ICPERMIT detects an expiration time after which the Interactive COBOL software will not be authorized to run, it will display a message to that effect.

When started with a valid protection method, ICPERMIT creates a failsafe security file, system.fp. The failsafe security file can be used to start ICPERMIT for up to 3 weeks after the failsafe security file was last written. (While ICPERMIT is running, the filesafe file is updated every 24 hours.) This insures that if the protection method fails, Interactive COBOL can still be used until you have time to get a replacement. From initial activation, the failsafe security file is good for 10 days. Within that time-frame you should insure that your protection method has been replaced.

If ICPERMIT gives the error: "There are no more license server connections available: Reserving slot in connection database." then you may need to use the Connection switch to increase your connection limit.

For local connections, ICPERMIT continuously checks to insure that the process that requested a license is still running. If it no longer sees the process, a message will be generated, and the license will be removed from use to be re-used. This check is not done for remote licenses. If a remote user terminates abnormally, that license will remain in use until ICPERMIT receives a terminate request via the TCP/IP keepalive mechanism.

Operation mode (-O a|c|d|e|i|m|p|t|u)

The Operation switch instructs ICPERMIT to perform some operation. All operation modes, except Check (-O c) require super user. Each Operation requires communication to a currently running ICPERMIT through the ICPERMIT TCP/IP socket. Valid options include: (a) amplify (turn on) tracing, (c)heck if already running; (d)isable licenses; (e)nable licenses; provide (i)nfo; (m) mute (turn off) tracing; (p) post connections; (t) terminate, (u) update icpermit. If no ICPERMIT is running an "Unable to establish a license server connection: The file was not found" will be given.

Operation amplify (-O a) sends an amplify command to an already running ICPERMIT to cause it to turn on tracing. If successful the following will be given:

```
Tracing of requests has been amplified.
```

Operation check (-O c) queries either the local or remote ICPERMIT to see if it is running. If successful, the pid and machine on which ICPERMIT is running is given. If successful, one of the following will be given:

```
The license server is running on pid 4294.
```

Or

```
The license server is running on pid 150 of computer servera.
```

Operation disable (-O d) sends a disable command to ICPERMIT to disable providing any new license authorizations. Licenses that are currently authorized will continue to operate but no new licenses will be granted. If successful the following will be given:

```
The software licenses have been disabled.
```

Operation enable (-O e) sends an enable command to ICPERMIT to re-enable any licenses that had previously been disabled. If successful the following will be given:

```
The software licenses have been re-enabled.
```

Operation info (-O i) queries ICPERMIT to provide current licensing information including licenses available, in use, and the most used. If successful the following type of message will be given:

```
License 1  (enabled): ICRUN - ICOBOL Runtime Revision 5.xx
  Serial# 00007006  OS: Any
  Authorized: 1 (max 513) of 3000    References: 515
License 2  (enabled): ICSQL - ICOBOL Integrated SQL Runtime Revision 4.xx
  Serial# 00007006  OS: Any
  Authorized: 0 (max 0) of 3000    References: 0
License 3  (enabled): ICDEV - ICOBOL Development Revision 5.xx
  Serial# 00007006  OS: Any
  Authorized: 0 (max 1) of 5    References: 1
License 4  (enabled): ICNET - ICOBOL Network Server Revision 5.xx
  Serial# 00007006  OS: Any
  Authorized: 0 (max 0) of 1000    References: 0
License 5  (enabled): ICSP2RUN - ICOBOL SP2 Runtime Revision 5.xx
  Serial# 00007006  OS: Any
  Authorized: 0 (max 0) of 1000    References: 0
```

Operation mute (-O m) sends a mute command to ICPERMIT to cause it to turn off tracing. If successful the following will be given:

```
Tracing of requests has been muted.
```

Operation post (-O p) sends the post command to ICPERMIT to instruct it to output to its audit log the current license connection information.

Operation terminate (-O t) specifies that the ICPERMIT service should be terminated.

Operation update (-O u) specifies that the ICPERMIT service should reread the license and file and reauthorize any additional licenses. This operation allows for licenses to be added or updated while the system is running.

The Disable license operation (-O d) and the Enable license operation (-O e) can be used to cause licenses to stop being provided without actually terminating ICPERMIT and to start allowing licenses to be provided again. This can be used to stop any new users from being allowed on without causing current users to be disconnected.

With the use of TCP/IP, ICPERMIT can be used over the Internet by allowing the ICPERMIT port to be accepted (through a firewall) by the Internet server site. Then remote users can use licenses from a single "master" license server at any time the Internet is available from their machine.

Licenses can be shared over a network by machines of the same or different flavors. (Windows to Linux or Linux to Windows).

When started with a valid protection method, ICPERMIT creates a failsafe security file, system.fp. The failsafe security file can be used to start ICPERMIT for up to 3 weeks after the failsafe security file was last written. (While ICPERMIT is running, the filesafe file is updated every 24 hours.) This insures that if the protection method fails, Interactive COBOL can still be used until you have time to get a replacement. From initial activation, the failsafe security file is good for 10 days. Within that timeframe, you should insure that your protection method has been replaced. The failsafe security file should NOT be modified in any fashion or else the file may NOT be usable when it is required. The failsafe security file should be tested routinely to insure that no system utility is modifying the file.

To use a failsafe security file, add the failsafe switch (-f) when starting ICPERMIT.

E.3. Notes

Licensed products do NOT know how they are licensed. They use the TCP/IP port created by an ICPERMIT to request authorizations.

F. ICPERMIT Termination

You must be super user to terminate ICPERMIT. The recommended method to terminate ICPERMIT is to use the appropriate system utilities.

For ln6 systems, the command is:

```
# service icobol stopicpermit
```

For ln7 & ln8 systems the command is:

```
# systemctl stop icpermit.service
```

However, ICPERMIT, can also be terminated by using the Terminate Operation (-O t) of ICPERMIT or by performing a KILL to the pid of the ICPERMIT process. The default KILL signal (SIGTERM) is all that is necessary to shutdown ICPERMIT and all Interactive COBOL processes. A SIGKILL (09) should never be used to terminate ICPERMIT except as a last resort when the software and/or system is otherwise hung.

ICPERMIT handles the following Linux signals with the given action:

SIGHUP (01)	- terminate ICPERMIT
SIGINT (02)	- terminate ICPERMIT if no Interactive COBOL processes are running
SIGQUIT(03)	- terminate ICPERMIT
SIGTERM(15)	- terminate ICPERMIT
SIGPWR (19)	- terminate ICPERMIT

When ICPERMIT terminates it will first prevent any new Interactive COBOL processes from starting and then it will terminate all running Interactive COBOL processes by first issuing a SIGTERM to each process and, if that does not stop the process after a brief delay, a SIGKILL (09) will be issued to the process. Thus, when ICPERMIT terminates, all Interactive COBOL processes should be gone and no new ones can start. Please be patient while ICPERMIT is terminating because it may take a significant amount of time to wait for various Interactive COBOL processes to terminate and to clean up various resources.

If ICPERMIT is ever terminated abnormally, (e.g., if a SIGKILL (09) was used to terminate it), its message key is left allocated (i.e., not cleaned up properly). Either the Linux system must be rebooted, the Linux *ipcrm* utility can be used to clean up these resources, or ICPERMIT can be restarted in which case it will clean up the resources properly and shutdown. You then must restart ICPERMIT again to actually bring ICPERMIT up.

On a system shutdown, the system shutdown script/executable will send the appropriate signals to ICPERMIT to cause it to start shutting down.

On shutdown, ICPERMIT will log its connection/license information to the log file.

IV. ICCONFIG

A. Introduction

ICCONFIG is used to create and edit configuration files (.cfi), terminal description files (.tdi), and printer translation files (.pti). Each of these files is a .ini based text file with the appropriate sections and definitions.

NOTE: Versions of ICOBOL before 3.30 used configuration files, terminal description files, and printer translation files with the extensions of .cf, .td, and .pt files that were binary files. This format has been discontinued. The ICREVUP utility (documented in the fromicobol3.txt file) has been provided to up-rev these files to their .ini-based counterparts. If you need to be able to configure these older files then you must keep an older copy of the configuration utility around to provide that ability. The current configuration utilities ONLY support the .ini-based format.

ICCONFIG provides a set of default values as a starting point for your configuration. However, every system will require tailoring to account for the requirements of the application, number of users, and system resources available.

FIGURE 4 is a summary of the ICCONFIG menu structure where the #n refers to the SCREEN number in this chapter:

```
MAIN MENU                                    #7
   - SYSTEM CONFIGURATIONS (.CFI)            #8
            * SYSTEM PARAMETERS               #9
            * ENVIRONMENT STRINGS             #10
            * CONSOLES AND PROGRAMS (@CONn)   #11
            * SERIALS (@SERn)                 #12
            * PRINTERS (@PRNn)                #13
            * PRINTER CONTROL QUEUES (@PCQn)  #14
            * PDF FORMATS                     #15
   - TERMINAL DESCRIPTIONS (.TDI)            #16
            * PARAMETERS                      #17
            * KEYBOARD                        #18
            * DISPLAY                         #19
            * COLOR/ATTRIBUTE MAPPING         #20
   - PRINTER TRANSLATIONS (.PTI)             #21
            * JOB CONTROL STRINGS             #22
            * CHARACTER MAPPING               #23
```

FIGURE 4. ICCONFIG MENU DIAGRAM

Generally we recommend going through each menu sequentially setting the appropriate values. You should understand your particular hardware configuration before getting to the devices.

B. Startup and Main Menu

To start ICCONFIG, the syntax is:

```
icconfig [-a[:aflag]|-A file|dir[:aflag]] [-b] [-h|-?] [-l|-L file|dir]
   [-O targetos] [-P ptname] [-q] [-T tdname] [ file ]
```

Where

-a[:*aflag*]|-A *file*|*dir[:aflag]* (Audit)
: Enables auditing (default icconfig.lg). Where *aflag* is a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-b (Batch)
: Enables batch mode operation.

-h|-? (Help)
Displays help text.

-l|-L *file*|*dir* (Load .ini)
Load the appropriate *file*.ini or *dir/file*.ini.

-O *targetos* (Set Operating system environment)
Specifies the default target operating system environment. If not specified, it defaults based on the current operating environment. Valid selections are Windows or Linux.

-P *ptname* (Set Printer translation)
Specifies the default printer translation entry. If not specified, it defaults to a one to one map. Valid selections can be seen with the Help switch.

-q (Quiet)
Enable quiet operation.

-T *tdname* (Set Terminal description)
Specifies the default terminal description entry. If not specified, it defaults based on the current operating environment as terminfo (On Linux) and pcwindow (On Windows). Valid selections can be seen with the Help switch.

file
Specifies the actual file to be configured. Files with the .cfi extension will go directly to the System Configuration menu, files with the .pti extensions will go directly to the printer translation menu, and files with the .tdi extensions will go directly to the Terminal Description menu. If no extension is given, the `.cfi' extension is added. For a simple file the file is sought only in the current directory.

If no argument is provided, ICCONFIG starts in the main menu.

If batch mode operation is specified, ICCONFIG starts and if the configuration file exists, it is read and if it was updated, it is rewritten and ICCONFIG terminates. If no configuration file exists, a default configuration file is created and ICCONFIG terminates.

If both batch mode and Set Terminal description are given, ICCONFIG does not use a configuration file but only acts on the given terminal description, just as batch mode alone acts on the configuration file. (Updates it if it exists and needs updating, otherwise creates one).

The -l|-L (Load .ini) switches (Batch update facility) are documented at the end of this section on page 74.

If the given configuration file exists and cannot be read at startup, an error is displayed and ICCONFIG terminates. The error should be fixed before re-running ICCONFIG. If the given configuration file does not exist at startup, a warning is displayed that the file was not found but ICCONFIG continues with the retrieved filename set to blank. If a save is done, the file will be saved under the startup name.

For example the line:

```
icconfig samplecf.cfi
```

would set the configuration file to be samplecf.cfi while in ICCONFIG. In the main screen of ICCONFIG, the file retrieved (if any) will be displayed and the filename that will be used on a Save or exit with update.

ICCONFIG uses the ICTERM environment variable to specify what terminal description to be used. If not specified, or if the specified entry is not one of the base default definitions known by ICCONFIG, terminfo will be used. To make full use of ICCONFIG, the terminal description should include 6 function keys, F1 through F6, and the arrow keys. If terminfo is being used, then the TERM environment entry must be set correctly for ICCONFIG to function properly.

While in ICCONFIG the general use of keys is:

1) the ESC key will exit from the current menu and return to the previous choice, in the MAIN MENU it will exit ICCONFIG.

2) up-arrow (↑) and down-arrow (↓) will move to the entry before or following the current entry. For menus that fit on one screen, at the top it will wrap to the bottom and at the bottom it will wrap to the top. For menus that scroll, at the top it will move to the bottom of the table and at the bottom it will move to the top of the table for up-arrow and down-arrow, respectively. The prompts will show these as <up> and <down> respectively since the arrow-characters are unprintable on most terminals.
3) function keys F1 and F2 will move to the previous field (left) and the next field (right) within the same row of a row-column table menu.
4) function keys F3 and F4 will move to the previous page (up) and the next page (down) keeping the cursor in the same location.
5) function key F5 will copy the fields for the current row or screen to the next row or screen. It is very useful in the initial setup to duplicate a standard entry down through a particular table. In the terminal keyboard configuration table setup it will insert a new entry.
6) function key F6 will delete an entry in the terminal keyboard tabe.
7) the Newline, Carriage-return, or ENTER key will select that choice and, for multi-field menus, move to the next entry.
8) the left-arrow and right-arrow keys act based on the type of field being entered.

Individual fields, within a menu, can be one of several types:

Yes/No fields accept Y, y, N, n, left-arrow, right-arrow, or **space** to select a value.
Ferris-wheel fields accept left-arrow, right-arrow, or **space** to select a value.
Data-entry fields, which can be either numeric or character, accept typed in values for the particular selection. Left-arrow and right-arrow in these fields just position within the field.
Menu fields accept either a number, up-arrow, or down-arrow to select a value.

SCREEN 7 shows the MAIN MENU displayed when ICCONFIG starts. Type the number for the selection to choose or use the up-arrow and down-arrow keys to change the value. Pressing ENTER for the currently displayed value executes that option.

```
      icconfig Revision 5.40 (Linux for x86 (ln7 64-bit))

        1.  Configure System Configurations (.cfi)
        2.  Configure Terminal Descriptions (.tdi)
        3.  Configure Printer Translations  (.pti)

       Selection:   __


   Press <up> or <down> to select, ESC to exit
```

SCREEN 7. ICCONFIG MAIN MENU

C. System Configurations (.cfi)

C.1. Overview

Option 1 from the MAIN MENU will display SCREEN 8, the System Configuration Menu. This menu provides the ability to configure each section of the system configuration files (.cfi).

```
       icconfig Revision 5.40 (Linux for x86 (ln7 64-bit))

         1.  Configure System Parameters
         2.  Configure Environment Strings
         3.  Configure Consoles and Programs (@CONn)
         4.  Configure Serial Lines (@SERn)
         5.  Configure Printers (@PRNn)
         6.  Configure Printer Control Queues (@PCQn)
         7.  Configure PDF Formats

         8.  Change Directory
         9.  Save Configuration File
        10.  Retrieve Configuration File
        11.  Reset Configuration to Defaults
        12.  Change Target OS Type

        Selection:
Directory:      /home/build
Retrieved file: system.cfi
Save(d) file:   system.cfi
Target OS type: UNIX

   Press <up> or <down> to select, ESC to exit
```

SCREEN 8. ICCONFIG SYSTEM CONFIGURATION (.cfi)

The Directory line shows the current directory for simple files.

The Retrieved file line displays the filename that was last retrieved, which may have been from the command line or selections. If no file was retrieved, this field will be blank and default values will be used for this configuration session.

The Save(d) file line displays the filename to which this session of ICCONFIG has been written; using selection 7, otherwise, it will be blank.

The *[target OS]* will show the target operating system for which this configuration is being setup for. If no configuration file is retrieved, it defaults to "UNIX" based on the current operating system unless ICCONFIG had been started with the Set Operating System switch (-O).

To start from the default case, type 11 to Reset the parameters to the defaults.

> **Note:** All the defaults listed in this manual and in ICCONFIG are what will be used on a Reset or if no configuration file can be found when ICCONFIG starts.

Pressing ESC will exit the screen and if ICCONFIG detects any modification since a Save or a Retrieve, it will prompt with a message asking if you wish to save your changes. If you type Y, a save is done before exiting; otherwise, your changes are discarded.

C.2. Configure System Parameters

Option 1 from the SYSTEM CONFIGURATION menu will show SCREEN 9. These parameters define how Interactive COBOL allocates various resources. Choose the appropriate values by typing a valid value for each parameter. Press ENTER to choose the displayed value and go on to the next parameter. Up-arrow and down-arrow can also be used to position to the previous or next menu selection. ENTER at the last field will position back to the top. Pressing ESC any time will return to the previous menu after checking to ensure that all parameters are consistent. If there is an error, the menu is re-displayed with the cursor on the offending entry. This entry must be corrected to exit the menu.

Terminal Status in Interactive COBOL will display the actual values that are current for some of these parameters along with the actual in use count for the files, record locks, and several other values.

System Information in Interactive COBOL will display the in use, maximum used, and configured values for many of these parameters during a particular invocation.

```
                System Parameter Configuration

Number of processes            32  Number of OPENs per process    128
Buffer area size (MB)           1  Buffer Write-thru enabled        N
Number of SEQUENTIAL files     32  Number of record locks         128
Number of ANSI INDEXED files   32  Number of ANSI Relative files    8
Enable 4GB ICISAM ver 7 files   Y  Create ICISAM Version 7 files    N

Printer Control enabled         Y  Printer control entries        48
Number of @CON devices         32  Number of @PCQ devices          32
Number of @PRN devices         32  Number of @SER devices          32

Number of PDF Formats        32

          Press <up>, <down> to position, ESC to exit.
```

SCREEN 9. SYSTEM PARAMETERS

Number of processes, is the maximum number of Interactive COBOL processes including ICEXEC, runtimes, utilities, compilers, and ICNETD servers that can execute simultaneously when ICEXEC is running. Valid entries are from 1 to 9999; 8 is the default.

Number of OPENs per process is the maximum number of simultaneous file opens that a single Interactive COBOL invocation (process) will allow to be opened (i.e., OPEN FD's in a COBOL program). The number of available handles is decreased every time a file is logically opened by any program. Valid entries are from 100 to 2048; 128 is the default.

Buffer area size (MB) is the amount of memory, in MegaBytes(MB), which ICEXEC will allocate in the shared memory area for buffers. If less memory is available, an error is returned by ICEXEC and it terminates. A certain minimum number of buffers must be available to keep from having a deadlock situation. The minimum buffer count is calculated as "maximum number of processes * 2", with the resulting minimum buffer area size being "minimum buffer count * 4KB". Each process has a local cache called the process local buffer cache that is calculated as:

```
process-local-buffer-cache = Buffer-area-size(MB) / Number-of-processes.
```

Generally this process local buffer cache should be such that it is much greater than 60KB, with larger values being better. When ICEXEC first starts, it displays the calculated process local buffer cache. Valid entries for *Buffer area size (MB)* are from 1 to 1024; 1 is the default.

For the best possible performance when writing ICISAM files the buffer size should be:

```
((max#ofkeysinafile * maxindexdepth) + 2) * 4KB/process
```

Thus, if an application has at most 4 alternates and no more than 3 levels of index then this formula would be:

```
((5 * 3) + 2) * 4KB = 68KB per process
```

If an application uses all 16 alternates and the maximum index depth of 6 then this formula would be:

```
((16 * 6) + 2) * 4KB = 416KB per process
```

This number is the process local buffer cache and a smaller value will cause the buffer cache to thrash on WRITE's as each key is inserted. ICEXEC determines the process local buffer cache by dividing Buffer size by number of processes. This number is reported when ICEXEC starts.

Buffer Write-through set to Yes instructs Interactive COBOL to write all modified pages to the operating system on any operation that modifies data that resides on the disk. Setting this option to Yes and configuring a very short flush time to Linux will keep modified pages flushed to disk at the expense of increased I/O overhead. Valid entries are Yes or No; No is the default.

Number of SEQUENTIAL files is the number of unique sequential files that can be simultaneously opened by the entire system. Valid entries are from 0 to 4096; 32 is the default.

Number of record locks is the number of simultaneous record locks allowed by the entire system. Valid entries are from 0 to 32767; 128 is the default.

Number of ANSI INDEXED files is the number of unique indexed files that can be simultaneously opened by the entire system. Valid entries are from 0 to 8192; 32 is the default.

Number of ANSI RELATIVE files is the number of unique relative files that can be simultaneously opened by the entire system. Valid entries are from 0 to 4096; 8 is the default.

Enable 4GB ICISAM version 7 files specifies whether when creating version 7 ICISAM files with the ability to have 4GB files. If not enabled (No) only 2GB ICISAM files are allowed. Valid entries are yes and no; Yes is the default. All version 8 ICISAM files allow more than 4GB.

Create ICISAM version 7 files specifies whether or not to create version 7 ICISAM files by default. If enabled (Y) version 7 files will be created. If not enabled (N) version 8 files will be created. The default is N. If backward compatibility with pre-5.00 systems is an issue, then this should be enabled (Y).

Printer Control enabled allows for the Printer Control Utility of Interactive COBOL to be enabled (Yes) or disabled (No). If set to No, files normally placed in the printer control file (system.pq) are not placed there and the IC_PRINT_STAT builtin will return an error. Valid entries are Yes or No; Yes is the default.

Printer Control entries is the maximum number of entries allowed in the printer control file at once. Once this number is reached, all new files will get a File Status 99 when a new file is being OPEN'ed which would create a new entry in the printer control file. Valid entries are from 48 to 1024 entries; 48 is the default.

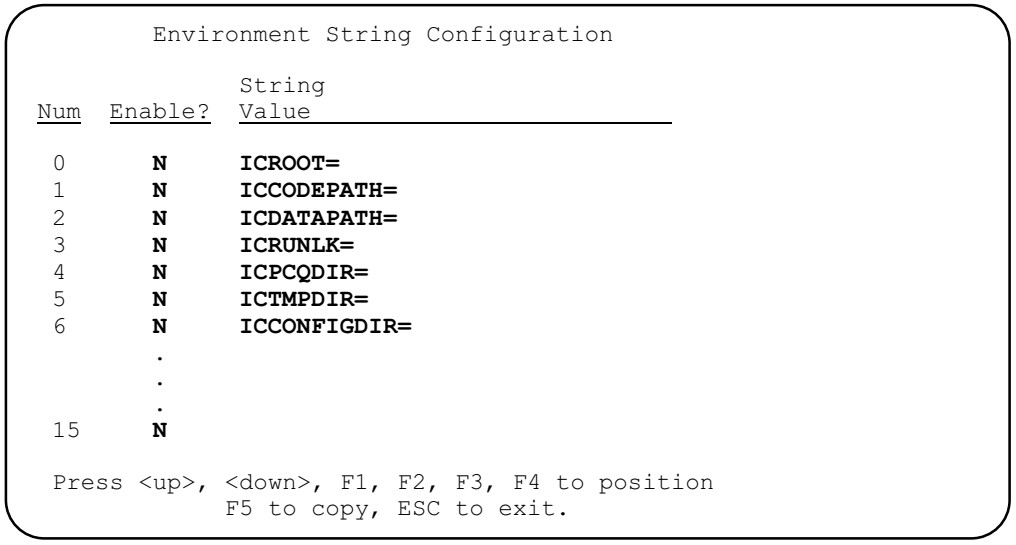
Number of @CON, @PRN, @SER, or @PCQ devices is the maximum number of each of those devices that you wish to configure. Valid entries are to 2048 entries; 128 is the default. These numbers will be used in the later configuration menus to set the maximum allowed set of devices. The minimum is 1 for @CONs and zero(0) for the others. You do not need to configure more than you actually need.

Number of PDF Formsts is the number of PDF Formats to be allowed to be configured in this configuration. Valid entries are from 0 to 256. 32 is the default.

C.3. Configure Environment Strings

Option 2 from the SYSTEM CONFIGURATION menu allows common environment entries to be specified. Entries defined in this section will be available to the runtime for all consoles. These common environment entries can be overridden by setting the same entry in the actual environment provided by the operating system.

Upon selecting this option SCREEN 10 is displayed.



SCREEN 10. ENVIRONMENT STRING CONFIGURATION

More on environment entries can be found on pages 81, 107 for Interactive COBOL on Linux.

Up to 15 default environment strings can be stored in the configuration file.

An environment string entry allows up to 255 characters.

C.4. Configure Consoles and Programs (@CONn)

Option 3 from the SYSTEM CONFIGURATION menu allows the configuration information for all the logical console lines (@CON0 and up) in the Interactive COBOL system to be defined. Upon selecting option 3, the CONSOLE and PROGRAM CONFIGURATION menu is displayed along with the current settings as shown in SCREEN 11. The actual number of entries is controlled by the Number setting in the System Parameters Configuration.

This menu allows a console to be enabled or disabled, an actual hardware device to be assigned to it, and any program setting can be defaulted.

```
        Console and Program Environment Configuration
                        Console number:   1

  Enable: Y          Device: ____________________
  Run programs? Y    Startup program:  ____________________________

  Console interrupt?             Y    Program debugging?          N
  Abort terminal?                N    System information?         Y
  Message sending?               Y    Printer control?            Y
  Printer control management?    N    System shutdown?            Y
  Terminal status?               Y    Detach/Host pgms?           Y
  Watch other terminals?         N    Exclude from being watched  N

                 Default Environment String Values
  ICTIMEOUT=  0       ICABORT= off           PCQ= 0  PRN= 0  SER= 0
  ICTERM= terminfo    ICLINES= 0             ICCOLUMNS= 0: 0
  ICSCROPT= full      ICSDMODE= disabled     ICREVERSE= process
  ICCOLOR= filter     ICBGCOLOR= black (0)   ICFGCOLOR= white (7)




 Press <up>, <down>, F3 previous, F4 next, F5 copy, ESC to exit.
Device: (blank), null, {char-device} in /dev, ip, machname, icrunrs
```

SCREEN 11. CONSOLE and PROGRAM ENVIRONMENT

The first entry is the console to be configured. Valid numbers are 0 to the highest supported console. Once selected, the actual hardware device specified for this console in the console configuration is displayed to the right of the *Run Program* entry along with whether the console is enabled or disabled.

Valid selections for each parameter and the defaults are:

Explanations:

Enable set to Yes allows this console to be used by Interactive COBOL.

Device, can be any of the following:

1) A character device available in the /dev directory. These will usually be in the form tty00, tty01, tty1a, tty1A, etc. . If the particular device name is not known, look in the /dev directory for the device name, or, if the device is a terminal, log on the terminal and use the *tty* command to get the device name. Do not enter the /dev prefix. If a device does not exist in the /dev directory when ICEXEC is started, a warning is given. This warning should only be viewed as an informational message if the device is a pseudo-device that gets created by the system when it is running. Otherwise, check to see if the name is really a valid device.

2) a valid *machine-name* or *ip-address* allows the console to be available for a ThinClient connection from that particular remote machine.

3) “icrunrs” allows the console to be available for ThinClient programs. These consoles are used on a first-come first-served basis unless the Terminal switch (-T) is used.

4) “null”, allows the console to be available for detached programs when using the IC_DETACH builtin. These consoles are used on a first-come first-served basis unless the Terminal switch (-T) is used.

5) “cgi”, allows the console to be available for cgi programs using icruncgi. These consoles are used on a first-come first-served basis. You should have enough cgi consoles to support the maximum simultaneous number of cgi connections you expect to have.

6) (blank), allows Interactive COBOL to use the console number to handle the following situations:

A) Starting Interactive COBOL with the Terminal number switch (-T). The console associated with the given "terminal-number" must have been configured with a blank device field.

B) Starting Interactive COBOL on a device that was not configured. If Interactive COBOL cannot find the device in the console table, it will scan from lowest to highest and use the first blank entry found that is not currently in use. It is useful to have several entries configured this way "just in case."

C) Starting a second copy of Interactive COBOL on a terminal. This can occur when calling Interactive COBOL from within Interactive COBOL using the CALL "|" feature. Since each Interactive COBOL must have a unique console number and the first Interactive COBOL already is using the one associated with the device, the second Interactive COBOL must find another console number to use. Again, it scans from lowest to highest to find the first one available.

D) Starting a ThinClient application. These consoles are used on a first-come first-served basis unless the Terminal switch (-T) is used.

The *machine-name* and/or *ip-address* allows for a specific machine to have a specific console number or numbers when using thinclients. The ip-address would be given as an *n.n.n.n* value.

Run Program specifies whether this console allows a COBOL program to be run on it. Valid selections are Yes or No. The default is Yes for @CON0 - @CON7 and No for all others.

Console lines with the Run program option enabled are called program lines. If the Run Program option is set to No this console device is treated like a serial device and all the remaining settings in this menu are ignored.

The number of program lines (or number of programs) works together with the Maximum number of processes and the licensed number of users to set an upper limit on the maximum number of programs that may be run simultaneously. This maximum number of programs is the lower of these three counts. The lower of the two configured values is shown in the Terminal Status and the System Information screens.

Startup program is the initial COBOL program to run when the runtime initializes this console. Valid selections are no entry or any valid COBOL program with up to 30 characters. The default is no entry, causing the COBOL program LOGON to be run.

Privileges Setup

These options provide for individual control over many system management functions about whether a COBOL program on this console has access to the particular feature. Valid selections are Yes or No.

Console interrupt privilege instructs Interactive COBOL whether to allow the user to abort the currently running COBOL program with the Linux Intr and Quit key sequences. If set to No, the character is passed on up to the program as data. When set to Yes, the Intr key set for this terminal is trapped by the runtime system and generates an abort to the currently running program and the Quit key for this terminal is trapped by the runtime system and generates a Hangup. The *stty isig* option is NOT used by Interactive COBOL - its initial setting is ignored. The runtime will set and clear the *isig* option as required.

Program debugging privilege: Allows the runtime to be started in debug mode. The default is No.

Abort terminal privilege determines whether COBOL programs running on this console may abort other terminals. The default is No except for CON0.

System Information privilege determines whether COBOL programs running on this console may view the system information screen.

Message sending privilege determines whether COBOL programs running on this console may send messages to other terminals.

Printer Control privilege determines whether COBOL programs running on this console may use the Printer Control Utility.

Printer Control Management determines whether this console is allowed to perform all operations on the printer control file while in the Printer Control utility. If this privilege is granted, this console may perform any operation on any file while in the printer control utility provided the user has access to the file from the operating system.

The *System Shutdown* privilege is ignored.

Terminal status privilege determines whether COBOL programs running on this console may access the terminal status screen or terminal control utility.

Detach/Host programs privilege determines whether to allow this program to detach COBOL jobs with the IC_DETACH builtin and, whether to allow the "|os-program" call for COBOL programs running on this console.

Watch other terminals privilege determines whether to allow this program to use the Watch Facility to Watch and/or Control another user. If enabled, the Watch and Control commands will be available to this user when in the Terminal Control Utility. If not enabled, the Watch and Control commands will NOT be available.

Exclude from being watched privilege determines whether to allow the program on this terminal to be Watched or Controlled by another terminal. If enabled, then this task will never be allowed to be watch'ed or controll'ed.

Default Environment String Values

This section specifies terminal specific information for a particular console line. The ICTIMEOUT, ICABORT, PCQ, PRN, SER, ICTERM, ICCOLUMNS, ICLINES, ICSCROPT, ICSDMODE, ICREVERSE, ICCOLOR, ICBGCOLOR, and ICFGCOLOR entries can be set in the user's environment to override any of these selections.

ICTIMEOUT sets a default global timeout value in seconds for all ACCEPTs and STOP literals on this console. If no key has been pressed in the specified time interval, the ACCEPT returns with the ESCAPE code set to 99. Valid selections are 0 through 6300; the default is 0 meaning no timeout, i.e., wait forever.

ICABORT instructs Interactive COBOL whether to abort the console (i.e., log it off) if an ACCEPT times out due to the global timeout setting (ICTIMEOUT). Valid selections are off or on; the default is off.

PCQ sets the generic printer control queue (@PCQ) to @PCQn based on the entered number. Valid selections are 0 through 2047; the default is 0. If set to an invalid queue, an error will occur on the OPEN.

PRN sets the generic printer device (@PRN) to @PRNn based on the entered number. Valid selections are 0 through 2047; the default is 0. If set to an invalid printer, an error will occur on the OPEN.

SER sets the generic serial device (@SER) to @SERn based on the entered number. Valid selections are 0 through 2047; the default is 0. If set to an invalid serial device, an error will occur on the OPEN.

ICTERM specifies the terminal description entry to be used for this console. The default for all consoles is terminfo. Valid ICTERM selections are valid terminal description entries with corresponding .TDI files. For an enabled console, this menu cannot be exited without some selection specified for ICTERM.

ICCOLUMNS and *ICLINES* set the number of columns and lines that will be allowed on this console. Valid selections are 0 through 255. The default of 0 says use the values specified in the terminal description entry. For terminfo and pcwindow descriptions, the defaults are those defined in the terminfo database and by the video bios hardware, respectively. These values indicate to Interactive COBOL where the screen wraps (ICCOLUMNS) and scrolls (ICLINES). If set incorrectly, screens may not display properly. The second selection for ICCOLUMNS is for compressed mode if supported by the terminal.

ICSCROPT specifies how the Interactive COBOL SCREEN OPTIMIZER is to perform. Valid selections are off, partial, full, and mute. The default is full. All enabled consoles have at least one screen area reserved. For a 24x80 column screen, a single screen image consumes about 8KB.

OFF says to transmit character codes as they are written by the program.
Partial enables the SCREEN OPTIMIZER to use a simple method of reducing the amount of characters sent to the terminal by comparing data to the single screen image.
Full enables the SCREEN OPTIMIZER to allocate an additional image of the current screen in memory and provides a complex method of comparing the data in the two images to reduce the amount of characters sent to the terminal to only those characters that would change the screen display at the end of an operation.
Mute forces Interactive COBOL to not send any implied codes to the terminal either at startup or termination. Only when executing a program instruction are codes sent to the terminal.

Ctrl-U from the keyboard while in an ACCEPT can be used to refresh the screen.

Usually partial and/or full will provide improved screen performance. Full is preferred with the debugger.

ICSDMODE specifies whether to enable the SCREEN HANDLER and if so in what mode. Valid selections are disabled, underline (0), reverse (1), and linedraw (2). The default is disabled. Linedraw uses the characters for line drawing specified in the terminal description file for the particular terminal.

ICREVERSE instructs Interactive COBOL how to interpret reverse codes from a COBOL program. Valid selections are filter, ignore, and process. The default is process. Reverse codes are Ctrl-B and Ctrl-V along with the two-byte sequences <036>D and <036>E.

Filter tells Interactive COBOL to watch for reverse codes from the program and to NOT send them to the terminal, since it does not support reverse.
Ignore tells Interactive COBOL that the user wants total control of the screen and may be sending binary reverse data to the screen and that Interactive COBOL should ignore all reverse codes (i.e., do not look for reverse codes). If running in this mode, the SCREEN OPTIMIZER cannot correctly repaint a user's screen that includes reverse codes.
Process tells Interactive COBOL to interpret reverse codes from the program and send the appropriate sequences to the terminal.

ICCOLOR instructs Interactive COBOL how to interpret color codes from a COBOL program. Valid selections are filter, ignore, and process. The default is filter.

Filter tells Interactive COBOL to watch for color codes from the program and to NOT send them to the terminal, since it does not support color.
Ignore tells Interactive COBOL that the user wants total control of the screen and may be sending binary color data to the screen and that Interactive COBOL should ignore all color codes (i.e., do not look for color codes). If running in this mode, the SCREEN OPTIMIZER cannot correctly repaint a user's screen that includes colors.
Process tells Interactive COBOL to interpret color codes from the program and send the appropriate sequences to the terminal. When set to Process, the initial background and foreground colors are set by Interactive COBOL at startup.

ICBGCOLOR sets the initial background color to the indicated value when running with ICCOLOR set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7). The default is black (0).

ICFGCOLOR sets the initial foreground color to the indicated value when running with Color support (ICCOLOR) set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7). The default is white (7).

Console lines that are not enabled are ignored. Console lines for which the hardware is not present give a warning at startup and an error on OPEN.

The table is searched from lowest to highest to decide the console-number for programs so the first occurrence of the device will select the console.

C.5. Configure Serial Lines (@SERn)

Option 4 from the SYSTEM CONFIGURATION menu defines the configuration information for the logical serial devices (@SER0 - @SER2047) in the Interactive COBOL system to be defined. Upon selecting option 4, the SERIAL CONFIGURATION menu is displayed along with the current settings as shown in SCREEN 12. The actual number of entries is controlled by the Number setting in the System Parameters Configuration. A serial device differs from a console device in that it can only be used for I/O. Console devices can be used to run programs or for I/O.

This menu defines logical serial devices to point to a particular hardware device.

```
        Serial Device (@SERn) Configuration

          @SER  Device in /dev          Enable?
             0  ____________________      N
             1  ____________________      N
             2  ____________________      N
             3  ____________________      N
             .
             .
             .
          2046  ____________________      N
          2047  ____________________      N

    Press <up>, <down>, F1, F2, F3, F4 to position,
              F5 to copy, ESC to exit.
```

SCREEN 12. SERIAL CONFIGURATION

Valid selections for each parameter and the defaults are:

Parameter	Valid Selections	Default
Device	Any character device in /dev	(blank)
Enable	Yes or No	No

Explanations:

Device can be any of the hardware character devices except for parallel printers. Two or more serial devices (@SERn) can point to the same hardware device.

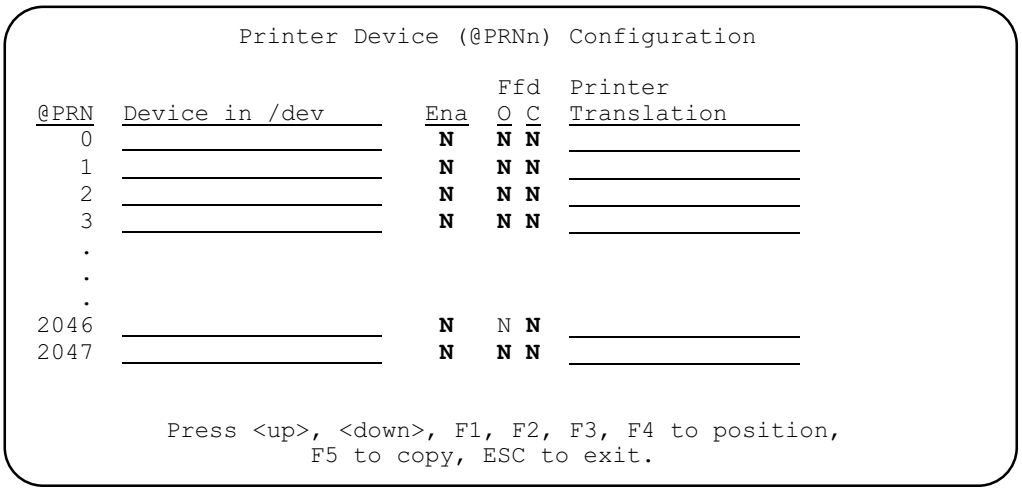
Enable set to Yes allows this serial device to be used.

Serial devices that are not enabled or for which the hardware device is not available are ignored and give an error on OPEN.

C.6. Configure Printers (@PRNn)

Option 5 from the SYSTEM CONFIGURATION menu defines the configuration information for logical printers (@PRN0 - @PRN2047) in the Interactive COBOL system to be defined. Upon selecting option 5, the PRINTER CONFIGURATION menu is displayed along with the current settings as shown in SCREEN 13. The actual number of entries is controlled by the Number setting in the System Parameters Configuration.

This menu defines certain characteristics for each logical printer to be specified.



SCREEN 13. PRINTER CONFIGURATION

Valid selections for each parameter and the defaults are:

Parameter	Valid Selections	Default
Device	Any character device in /dev	(blank)
Enable	Yes or No	No
FF on OPEN	Yes or No	No
FF on CLOSE	Yes or No	No
Printer Translation	filename	(blank)

Explanations:

Device directs Interactive COBOL where to send the print data for a particular logical printer. Possible selections are any of the hardware character devices that are not in use as terminal lines or blank for None. Two or more printers (@PRNn) can point to the same hardware device.

Enable set to Yes allows this printer device to be used.

FF on OPEN and *FF on CLOSE* instructs Interactive COBOL whether to send a Form-Feed to the printer when the appropriate statement is executed on a particular printer.

Printer Translation specifies a printer translation file to be used when printing. If nothing is specified, each character is printed as is. Printer translation files are opened and read when ICEXEC is started.

Printers that are not enabled or for which the hardware device is not available, are ignored and give an error on OPEN.

PRN devices are direct linkages from the Interactive COBOL process to the device. When used with the COBOL EXCLUSIVE option, Interactive COBOL prevents other Interactive COBOL processes from opening that device while it is in use.

There is no standard way to keep a normal Linux process from also opening a device opened by Interactive COBOL, since there is no exclusive option on a Linux open. Therefore, it is generally better to use PCQs rather than PRNs. PCQs print though the Linux print spooler that has its own locking mechanism to coordinate access to the printer.

C.7. Configure Printer Control Queues (@PCQn)

Option 6 from the SYSTEM CONFIGURATION menu defines the configuration information for up to 2048 printer control queues (@PCQ0 - @PCQ2047) in the Interactive COBOL system to be defined. Upon selecting option 6, the PRINTER QUEUE CONFIGURATION menu is displayed along with the current settings as shown in SCREEN 14. The actual number of entries is controlled by the Number setting in the System Parameters Configuration.

This menu defines certain characteristics for each logical printer control queue to be specified.

```
      Printer Control Queue (@PCQn) Configuration

                                 Printer      Aut End       No
@PCQ    Print Queue     Enb      Translation  Que Disp  Ntf Ban
   0                    N                     N   Keep    N   N
   1                    N                     N   Keep    N   N
   2                    N                     N   Keep    N   N
   3                    N                     N   Keep    N   N
   .
   .
   .
2047                    N                     N   Keep    N   N

    Press <up>, <down>, F1, F2, F3, F4 to position,
              F5 to copy, ESC to exit.
```

SCREEN 14. Linux PRINTER QUEUE CONFIGURATION

Valid selections for each parameter and the defaults are:

Parameter	Valid Selections	Default
Print Queue	Any valid *lp* destination	(blank)
Enable	Yes or No	No
Printer Translation	filename	(blank)
Auto Queue	Yes or No	No
Disposition	Keep, Remove, or Delete	Keep
Notify	Yes or No	No
No banner	Yes or No	No

Explanations:

Print Queue directs Interactive COBOL where to send the data for a particular logical printer control queue. Possible selections are any of the valid destinations known to the Linux print spooler. The *lpstat* utility can be used to view the current selections available. A blank entry uses the default destination. The given destination is used as the destination field in an *lp* command. For example,

lp -ddest file-to-print.

A destination must be accepting entries for the files to be printed.

Enable set to Yes allows this printer control queue device to be used.

Printer Translation specifies a printer translation file to be used when printing. If nothing is specified each character is printed as is. Printer translation files are opened and read when ICEXEC is started.

Auto Queue instructs the Printer Control utility whether to automatically queue a file to its default print queue when the file has been closed.

Ending Disposition provides the Printer Control utility with the default option for a particular print file of whether to Keep, Remove, or Delete the particular file or entry after it has been printed.

Notify instructs the Printer Control utility whether to instruct the Linux print spooler to send a message to the user when the file has finished printing. The is usually done with the "-w" option to *lp*.

No banner instructs the Printer Control utility whether to set the nobanner option when sending the file to the Linux print spooler. The default is No, i.e., do NOT set the nobanner option. For the nobanner option to work the shell script that prints the file must support the nobanner option as Interactive COBOL sets it. That is, the shell script must support the option as `-o nobanner' on the *lp* command.

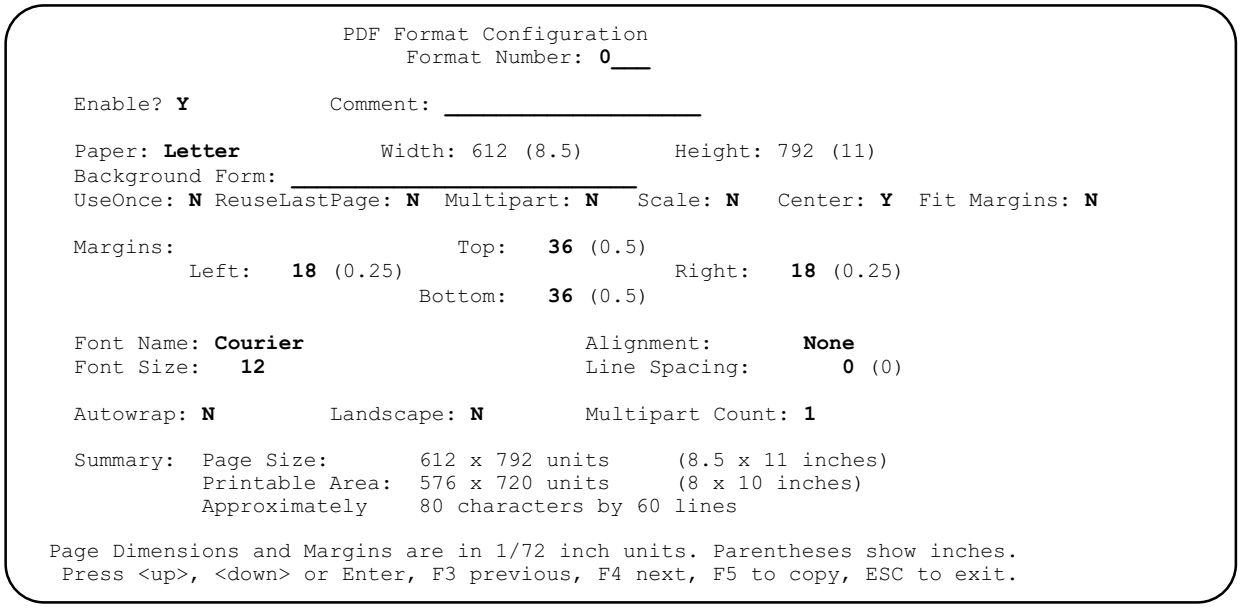
Printer control queues that are not enabled in ICCONFIG are ignored and give an error on OPEN.

Printer control queues (@PCQs) are indirect linkages from Interactive COBOL to a particular device through the Linux print spooler. The data is spooled to a disk file and when the file is closed under COBOL the disk file is submitted to the appropriate print spooler to be printed on the indicated print device. If a printer control queue is opened directly, e.g., "@PCQ25", Interactive COBOL provides intercept spooling. Intercept spooling is the capturing of all subsequent data to a temporary file and on a CLOSE to queue the temporary file to the appropriate printer control queue. For intercept spooling, Interactive COBOL pipes the output data to the *lp* command with the destination set to that specified above.

C.8. Configure PDF Formats

Option 7 from the SYSTEM CONFIGURATION menu defines the configuration information for up to 256 PDF Formats. Upon selecting option 7, the PDF FORMAT menu is displayed along with the current settings as shown in SCREEN 15. The actual number of entries is controlled by the Number setting in the System Parameters Configuration.

This menu defines certain characteristics for each PDF Format to be specified.



SCREEN 15. PDF FORMATS CONFIGURATION

Where:

Format Number is the particular format description to be configured. The range of values is as specified in the System Parameters configuration. Currently at most 256 formats can be specified.

Enable set to Yes allows this PDF Format to be used.

Comment provides an optional brief description of this format. This description is stored in the shared area and is viewable by ICSMVIEW or in the Printer Control Utility.

Paper allows for a particular paper size to be entered by scrolling through the various predefined sizes or by entering a custom size which allows for the specific size to be set. Valid selections include A5, A4, Executive, Lineprinter, Tabloid, Ledger, Legal, Letter, and Custom. Letter is the default.

Background Form is optional and allows a background image file (in .pdf format) to be specified that will be imposed on the pdf image to be created. At runtime this form must be present in the ICCONFIGDIR directory or the current working directory.

The following selections only apply if a background image is specified:

UseOnce: **Y/N** (Default is N)

Y=use the background form one time (ReuseLastPage appears)
N=old behavior (MultiPart appears as before)

ReuseLastPage: **Y/N** (Appears only when UseOne=Y)

Y=once the form has been used, reuse the last page of the form for all the remaining pages.
When used with a 1-page form, the effect is the same as UseOnce=N.
N=once the form is used, print remaining pages with no form.

ReuseLastPage is useful to have a unique first page.

Multipart specifies whether this background image is a multi-page document. When set, the runtime will generate a logical page multiple times for each page in the image. The default is No.

Scale specifies whether to scale the background image. When set to Yes, the image will be scaled to either the paper size or margin size. The aspect ratio is kept intact. The default is No.

Center specifies whether the background image should be centered. When set to Yes, the image will be centered to either the paper size or margin setting. The default is Yes.

Fit Margins specifies whether to use the margin settings or the paper size should be used when scaling or centering the image. The default is the No (use paper size).

Margins allows the specific inside margins (Top, Left, Right, Bottom) to be specified for this format. Units are in points, which are 1/72 inch units. The defaults are 36 (.5 in) for Top and Bottom and 18 (.25 in) for Left and Right.

Font Name allows for a specific supported font to be entered. Currently supported fonts include: Courier, Courier-Oblique, Courier-Bold, Courier-BoldOblique, Helvetica, Helvetica-Oblique, Helvetica-Bold, Helvetica-BoldOblique, Times-Roman, Times-Italic, Times-Bold, and Times-BoldItalic. These are 12 of the 14 standard Adobe fonts. The default is Courier.

Alignment is shown when a proportional font is specified and instructs how characters are to be placed on a page. Valid selections are: None, Character, Word. None is the default. Character will treat the font like a fixed font and place each character is a fixed position on the line. Word will set each word into a calculated fixed position.

Font Size is the specified size in points. Sizes from 2 - 72 can be specified. For example, a 12-point Courier font provides 10 characters per inch. The 10 is usually referred to as the pitch for fixed fonts. A 10-point Courier provides 12 characters per inch, i.e., 12 pitch. The default is 12.

Line Spacing specifies the default spacing between lines in points. The line height is the sum of *font size* and *line spacing*. The default is 0.

Autowrap specifies whether to wrap lines that are too long or truncate the lines. The default is No (truncate).

Landscape specifies whether this page should be treated as landscape or portrait. (Swaps width and height.) The default is No (portrait).

Multipart Count specifies whether to generate multiple pages for each page. If this value is greater than 1, then a multi-part form will be generated. If a multi-page image is specified and multipart is set to Y then this value MUST match the number of pages in the image. If a multi-page image file was specified, but the multipart is set to N, then this file will be used in a modulo fashion as logical pages are presented. The default is 1.

The Summary section at the bottom of the screen shows a summary of the pdf page specifics. This is kept constantly updated as selections are made in the screen.

All page dimensions and margins are in points which are 1/72 inch units. The values in parenthesis (x) show inches.

C.9. Change Directory

The Change Directory selection from the SYSTEM CONFIGURATION menu allows the current directory to be changed.

C.10. Save

The Save selection from the SYSTEM CONFIGURATION menu allows any changes of the current values in the configuration file to be saved. You are prompted on the Save(d) file line for the filename (without the `.cfi' extension) to which this session should be saved. The default will be the retrieved name or system.cfi. If no Save is done, then any changes made are NOT saved. If you try to exit and no Save has been done since the configuration was last modified, you are prompted whether a Save should be done.

> **Note:**
> Once a Save is done with new values, ICEXEC will not see the changes until the next time ICEXEC is executed. I.E., you cannot change the configuration in effect while ICEXEC is running, although the configuration file can be modified.

C.11. Retrieve

The Retrieve selection from the SYSTEM CONFIGURATION MENU allows a configuration file to be read such that it can be viewed or updated. The filename to be retrieved is prompted for on the Retrieved file line.

C.12. Reset to Defaults

The Reset to Defaults selection from the SYSTEM CONFIGURATION MENU allows you to start from scratch and set all new values or just to see the default configuration values. Remember you do not have to Save the results of a work session in ICCONFIG.

C.13. Charge Target OS

Allows the target OS type to be changed to Windows or UNIX (Linux) .

D. Terminal Descriptions (.tdi)

D1. Overview

Option 2 from the MAIN MENU will display SCREEN 16. This menu provides the ability to configure different terminal types for ICTERM. The terminal descriptions are saved to a file called the terminal description file with a `.tdi' extension. To build a terminal description file, use the menu to select the appropriate base terminal (selection 1) and then use options 2 thru 6 to customize the configuration and finally use Save (selection 8) to create that description.

The name of the current base terminal selection is displayed in this screen and the Configure Keyboard and Configure Display screens.

```
      Terminal Description (ICTERM) Configuration
      Name: terminfo
      Base: terminfo           Comment: terminfo (UNIX)

       1.  Select Base Terminal
       2.  Change Comment
       3.  Configure Parameters
       4.  Configure Keyboard
       5.  Configure Display Characters
       6.  Configure Color/Attribute Map (pcwindow)

       7.  Change Directory
       8.  Save Terminal Description File
       9.  Retrieve Terminal Description File

       Selection:   1

Directory:      [directory]
Retrieved file:
Save(d) file:


       Press <up>, <down> to select, ESC to exit.
```

SCREEN 16. ICCONFIG TERMINAL DESCRIPTION (ICTERM) (.tdi)

To allow a particular terminal description to be available for Interactive COBOL, a terminal description file (.TDI) must be created for that particular ICTERM entry. Each of the base terminal descriptions can be used to create a customized description file. This can be done by using option 1 (Select Base Terminal) followed by the needed coniguration options and finally option 8 (Save terminal description file) for each of the needed customized files.

All the base terminal descriptions are available by default within the runtimes without needing the actual file. Only customized files need to be provided.

To change a terminal description file, use option 1 (Select Base Terminal) or option 9 (Retrieve Terminal description file) to choose the description to be changed. Now use options 4 (Configure Keyboard), 3 (Configure Parameters), and/or 5 (Configure Display Characters) to change the needed values and then use option 8 (Save Terminal Description file) to store that terminal description. Option 2 (Change comment) can be used to store a comment about this new description.

To build a new terminal description file use option 1 (Select Base Terminal) or option 9 (Retrieve Terminal description file) to choose a description whose output characteristics match the new terminal. Now use options 4

(Configure Keyboard), 3 (Configure Parameters), and/or 6 (Configure Display Characters) to change the needed values for the particular terminal and then use option 8 (Save Terminal Description file) with a new filename. These tailored versions should generally be saved with a different name and comment to distinguish them from the defaults. Option 2 (Change comment) can be used to store a comment about this new description.

Option 6 (Configure Color/Attribute Map) is only for use with the pcwindow base terminal and allows the Windows master console screen to be setup.

D.2. Select Base Terminal

Option 1 from the TERMINAL DESCRIPTION menu positions to the Base Terminal Description field and allows a base terminal to be selected from a list of available terminals. These can be selected via a ferris wheel using the left-arrow and right-arrow keys. This base set includes the following:

Ansi class:	ANSI, AIX Console, AT&T 605, AT386 Console, SCO Console, SUN Console, XENIX Console, XTERM Console, 386IX Console, Apple Mac Terminal
DG class:	DG D200+
DGUNIX class:	DG D217+ (in dgunix mode)
Disk class:	File
IBM class:	IBM 3151
Freedom class:	FREEDOM-One
VTxxx class:	VT100, VT220, VT220pc, Linux
Wyse 50 class:	WYSE 50
Wyse class:	WYSE 60
PC Window class:	PCWINDOW, PCWINDOWCOLOR, PCWINDOWMONO (On Windows only)
Terminfo class:	TERMINFO (On Linux only)

The sets of classes shown above are the default output codes used for each of the particular base terminal descriptions. I.E., all the terminal descriptions in the Ansi class use ansi codes, those in the DG class use DG control codes, etc. . When in the Configure Display screens, these classes are shown at the top as Format.

More information about these base terminal descriptions can be found in the ICTERM Chapter on page 125.

D.3. Change Comment

Option 2 from the TERMINAL DESCRIPTION menu can be used to insert a comment into a terminal description file. Selecting option 2 positions to the Comment field allowing up to thirty (30) characters to be entered for this terminal description. This option is especially useful if you change a base terminal description to provide some documentation of the change. For example if you add a HotKey to a DG description you may change to comment to "DG F1 Hot".

D.4. Configure Parameters

Option 3 from the TERMINAL DESCRIPTION menu gives the PARAMETER CONFIGURATION as shown in SCREEN 17.

The current name of the terminal being configured is displayed at the top of the menu along with its format. The format can be any of the valid display classes mentioned previously.

The PARAMETER CONFIGURATION provides the basic defaults for this terminal description for the number of lines and columns and what characters to use for the line drawing character set. The lines and columns values can be overridden by the ICLINES and/or ICCOLUMNS environment entries either in the actual environment or in the PROGRAM ENVIRONMENT screen specified previously in ICCONFIG.

```
            Parameter Configuration
    Name: terminfo
    Base: terminfo           Comment: terminfo (UNIX)

                   Keyboard
           Sequence Timeout (.1 sec): 5

                  Screen Size
Lines:   24   Normal Columns:  80    Compress Columns:   80

           Line Drawing Character codes
Upper Left Corner:      43    Upper Right Corner:     43
Lower Right Corner:     43    Lower Left Corner:      43
Horizontal Line:        45    Vertical Line:         124


  Press <up> or <down> to position, ESC to exit.
```

SCREEN 17. PARAMETER CONFIGURATION

The sequence timeout is the number of tenths of seconds to wait for the intercharacter gap between function keys sequences. The default is 5. Some remote connections, using telnet or rlogin, may need to have this number adjusted upward to correctly handle function key sequences.

The screen size and line drawing codes are NOT available for terminfo type terminals.

The screen size specifies the default rows and columns for this terminal type. If the Compress columns is set to a value other than that set for normal columns then Compressed mode is supported on this terminal.

The line drawing codes are the decimal codes for the ASCII character to be used for each appropriate part of a box. The default values shown are what this terminal supports. If the terminal does not support a real line drawing character set, the default values of 43, 45, and 124 (decimal) which are `+', `-', and `|', are used.

D.5. Configure Keyboard

Option 4 from the TERMINAL DESCRIPTION menu presents the screen as shown in SCREEN 18. The KEYBOARD CONFIGURATION menu instructs Interactive COBOL what to do for each possible input sequence from this keyboard.

The current name of the terminal being configured is displayed at the top of the menu along with its format. The format can be either ASCII or terminfo. In ASCII mode, this table instructs Interactive COBOL how to handle each of the entered ASCII sequences on input from the keyboard. In terminfo mode, this table instructs Interactive COBOL how to handle each of the entered terminfo capname sequences on input from the keyboard.

This table can have up to 512 unique input sequences for a particular terminal description.

The first column of the table gives the legend or label of the key and any shift or ctrl key that must be pressed to get this key. Interactive COBOL does not use this column in any way, it is useful only as a label.

Column 2 gives the actual codes (series of 8-bit bytes) that this key generates when pressed on the keyboard. Interactive COBOL normally watches for these code(s) in a timing-sensitive fashion and, if seen, will handle as specified by the next field.

When entering data in column 2 the following can be used:

a enters a bell (Ctrl-G)
b enters a backspace (Ctrl-H)
dnnn enters nnn in
e enters an ESC
f enters a form-feed (Ctrl-L)
k allows a terminfo Capname to be entered when using terminfo
n enters a linefeed (Ctrl-J)
r enters a carriage-return (Ctrl-M)
t enters a tab (Ctrl-I)
v enters a vertical tab (Ctrl-K)

xnn enters nn in hex
\\, \^, \″, \′, and *?* enters a single `\',
`^', `"', and `?' respectively

^@ through ^_ enters the control
code (\000 - \037)
nnn enters nnn in octal
any printable character (`!' - `~') is
entered as itself

When displayed on the screen or in the listing, all printable characters (`!' - `~') will be displayed as is, while all others will be shown in octal.

The next columns allow the type to be entered; within that type, how to interpret the key (code), and for next, previous, and terminating types whether to erase to the right of the cursor position. Right-arrow or left-arrow is then used to cycle forward or backward through the valid choices through these ferris-wheel fields.

The up-arrow, down-arrow, F1 (left), F2 (right), F3 (page-up), and F4 (page-down) keys will position to the field to change.

```
                    Keyboard Configuration
      Name: terminfo
      Base: terminfo           Comment: terminfo (UNIX)

  Label   Byte String  Ext Type             Code           Erase
  Ctrl-A  \001          N  Editing Function  End-of-field     N
  Ctrl-B  \002          N  Editing Function  Left Word        N
  Ctrl-E  \005          N  Editing Function  Ins mode ON/OFF  N
       .
       .
       .
 Press <up>, <down>, F1-F4 position, F5 copy, F6 delete, ESC exit.
```

SCREEN 18. KEYBOARD CONFIGURATION

This table is always stored in sorted order based on the input sequence defined in column 2.

Valid Types with their Codes are defined in the following paragraphs. The numbers in parentheses after the Types and Editing codes are the values returned in the IC_GET_KEY builtin function for the appropriate keystroke.

Normal Character (1) - The runtime system will treat keycodes of this type as normal 8-bit ASCII characters. The Code column contains the actual character code to be used by the runtime system. The value can be 0 to 255 (decimal).

Editing Function (2) - This type of keycode instructs the runtime system to take the action as described in the Code column. The possible actions in the Code column are:

clear to end-of-field (7)	beginning of field (8)	end of field (9)	right word (10)
left word (11)	destructive TAB (12)	left TAB stop (13)	right TAB stop (14)
left a character (1)	right a character (2)	backspace (3)	delete a character (4)
insert mode ON/OFF (5)	clear field (6)	sound bell (15)	back delete (16)

TAB settings are set at every fourth character position from the beginning of the field, i.e., 1, 5, 9, ...

Terminate Field (3) - This type of keycode causes the runtime system to accept the current field and set the ESCAPE KEY value to the value given in the Code column. The ESCAPE KEY value can be 00 to 99, but remember that value 99 is used for timeouts.

Previous Field (4) - This type of keycode instructs the runtime system to move to the previous field in a screen. If the current field is not the first field in a screen, the field is accepted and the screen is positioned to the preceding field. If the current field is the first (or only) field in a screen, the result depends on the ESCAPE KEY value associated with the key given in the Code column. If the ESCAPE KEY value is 00 (default), the system will beep, and the screen will remain positioned at the first field. If the ESCAPE KEY value is not 00, the field will be

accepted, and the screen will exit with the specified ESCAPE KEY value. The ESCAPE KEY value can be 00 to 99, but remember that the system returns 99 for timeouts.

Next Field (5) - This type of keycode instructs the runtime system to move to the next field in a screen. If the current field is not the last field in a screen, the field is accepted and the screen is positioned to the next field. If the current field is the last (or only) field in a screen the field will be accepted and the screen will exit with the specified ESCAPE KEY value given in the Code column. When using the default value, it will act just like a newline had been hit. The ESCAPE KEY value can be 00 to 99, but remember that the system returns 99 for timeouts.

Use of the *Previous Field* and *Next Field* functions on the up-arrow and down-arrow keys with unique ESCAPE KEY values will allow applications to tie several screens together and control the flow from one screen to the other. The default setting of up-arrow is Previous field-ESCAPE KEY 00. The default setting of down-arrow is Next field-ESCAPE KEY 00.

Hot Key Function (6) - Allows for a particular hotkey program to be called whenever this key is entered. The code column allows a value from 00 to 99 to be set such that a COBOL CALL "hotkey<nn>" will be called with the given value replacing the <nn>. There must be a COBOL program available and executable with this name or else a beep will be given.

Previous Row (7) - This type of key code instructs the runtime system to move to the "best fitting" field on a previous row in a screen. If the current field is not in the topmost row of the screen, it is accepted and the cursor is positioned to the "best fitting" field. If the current field is in the topmost row of the screen, the result depends on the ESCAPE KEY value associated with the key in Code Column in ICCONFIG. If the ESCAPE KEY value is 00, the screen will remain positioned in the current field and the bell will sound. If the ESCAPE KEY value is not 00, the field will be accepted and the screen will exit with the specified ESCAPE KEY value. The "best fitting" field is defined to be a field in a preceding row which has the same column position (1st choice), a higher column position (2nd choice) or a lower column position (last choice) than the current field. In any case, the field selected will be the first screen row proceeding the current one which contains ANY fields.

Next Row (8) - This type of key code instructs the runtime system to move to the "best fitting" field on a subsequent row in a screen. If the current field is not in the bottommost row of the screen, it is accepted and the cursor is positioned to the "best fitting" field. If the current field is in the bottommost row of the screen, the result depends on the ESCAPE KEY value associated with the key in Code Column in ICCONFIG. If the ESCAPE KEY value is 00, the screen will remain positioned in the current field and the bell will sound. If the ESCAPE KEY value is not 00, the field will be accepted and the screen will exit with the specified ESCAPE KEY value. The "best fitting" field is defined to be a field in a subsequent row which has the same column position (1st choice), a lower column position (2nd choice) or a higher column position (last choice) than the current field. In any case, the field selected will be the first screen row following the current one which contains ANY fields.

Special Function (0) - Is a set of special internal actions to be taken by the runtime upon receipt of this keystroke. *Special Function* keys do not return in IC_GET_KEY. The actions are defined by the following Codes:

Illegal Character - The runtime system will beep when it receives a keystroke of this type.

Ignored Character - The runtime system ignores keycodes of this type.

Refresh Screen - The runtime system will clear the current screen and totally refresh the screen from its internal image.

Enter minus - This runtime system enters a minus character key followed by a *Terminate Field* with an ESCAPE KEY 0, as two separate keystrokes.

The Erase column is only valid for *Terminate Field*, *Next Field*, *Previous Field*, *Previous Row*, and *Next Row* types. If Erase is set to No (the default), the runtime accepts the current field as currently displayed. If set to Yes, all characters to the right of the cursor in the current field are discarded. It is equivalent to first entering the clear to end-of-field key followed by the same Terminate, Next, or Previous key without the Erase option.

When configuring for Linux and using the terminfo base setting, Terminfo Capname codes should be entered by preceding the Capname with a backslash (\). For example, *\kcud1* would be entered for cursor down.

The Ext column is the timing-insensitive option and can be used to configure multi-character keystroke sequences for those terminals that do not support the needed number of function keys. The lead-in character for a timing-insensitive sequence must not have been previously defined as a timing-sensitive character, otherwise an error is given by ICCONFIG. An example of how timing-insensitive keys can be entered is given below:

Let's say you want the Ctrl-R character to be the lead-in character for your timing-insensitive codes and you wish to use Ctrl-R followed by the `1' key for function key F1, followed by a `2' for F2, and so on up to `9', Ctrl-R followed by an `a' for F10, `b' for F11, . . . , `f' for F15 and the shifted states of the second character to get the shifted function keys. I.E., Ctrl-R followed by `!' for Shift-F1, and Ctrl-R followed by `A' for Shift-F10 etc. .

Now, to allow the Ctrl-R character to be the lead-in for the functions keys F1-F15 in the normal and shift states and allow the Ctrl-R Ctrl-R keystrokes to be interpreted as a single Ctrl-R, change the default Ctrl-R Byte String entry from "\022" to "\022\022" with the Ext column set to Y.

Now for each needed function key add the appropriate line. For the first case it would be:

In column 1 (Label), give an appropriate label:
```
MyF1
```

In column 2 (Byte String), give the string:
```
\0221
```

In column 3 (Ext), enable the key:
```
Y
```

In column 4 (Type), give the type code as:
```
Terminate Field
```

In column 5 (Code), give the appropriate function key code to be returned:
```
ESCAPE 2
```

In column 6 (Erase), give the appropriate value on whether to erase to the right of the cursor:
```
N
```

Now continue these responses row by row to add the needed keys as given below:

```
\0222    to generate F2
.            .
\0229    to generate F9
\022a    to generate F10
.            .
\022f    to generate F15
\022!    to generate Shift-F1
.            .
\022(    to generate Shift-F9
\022A    to generate Shift-F10
.            .
\022F    to generate Shift-F15
```

Additional timing-insensitive keys can be added by following the above example.

D.6. Configure Display Characters

Option 5 from the TERMINAL DESCRIPTION menu presents the screen as shown in SCREEN 19. The DISPLAY CHARACTER CONFIGURATION instructs Interactive COBOL what to do for each possible output character from the COBOL program to the terminal. This option is not available for terminfo.

The current name of the terminal being configured is displayed at the top of the menu along with its format.

This table provides entries for all 256 possible sequences with each line representing an entry as two major columns: Character from program and Character to Display.

The first major column, Character from Program, gives the character from the program to be output to the terminal in Decimal, Octal, Hex, and as a Description.

The second major column, Character to Display, give a Value column that allows entry of the value to be displayed along with 5 additional sub-columns showing the Decimal, Octal, Hex, Description, and what this output character looks like on this terminal.

```
              Display Character Configuration
             Name:    dg
             Base:    dg          Comment: DG D200+

  Character from Program      Character to Display
  Dec Oct Hex Description     Value  Dec Oct Hex Description Chr
  --- --- --- -----------     -----  --- --- --- ----------- ---
    0 000 00  Ctrl-@          \000     0 000 00  Ctrl-@      N/D
    1 001 01  Ctrl-A          \001     1 001 01  Ctrl-A      N/D
    2 002 02  Ctrl-B          \002     2 002 02  Ctrl-B      N/D
    3 003 03  Ctrl-C          \003     3 003 03  Ctrl-C      N/D
      .
      .
      .
  254 376 FE  \376            \376   254 376 FE  \376        ( )
  255 377 FF  \377            \377   255 377 FF  \377        ( )

Press <up>, <down>, F1-F4 to position, F5 to copy, ESC to exit.
```

SCREEN 19. DISPLAY CHARACTER CONFIGURATION

When entering data in the Values column the following can be used:

a enters a bell (Ctrl-G)	*t* enters a tab (Ctrl-I)
b enters a backspace (Ctrl-H)	*v* enters a vertical tab (Ctrl-K)
dnnn enters nnn in decimal	*xnn* enters nn in hex
e enters an ESC	\\, \^, \″, \′, and *?* enters a single `\', `^', `"', `", and `?' respectively
f enters a form-feed (Ctrl-L)	^@ through ^_ enters the control code (\000 - \037)
n enters a linefeed (Ctrl-J)	*nnn* enters nnn in octal
r enters a carriage-return (Ctrl-M)	any printable character (`!' - `~') is entered as itself

When displayed in the Chr column, all non-control code characters (both 7- and 8-bit) will be displayed as defined within parenthesis ( ), while the control code characters will show a N/D (Not Displayable) in that column.

D.7. Configure Color / Attribute Map (pcwindow types)

Option 6 from the TERMINAL DESCRIPTION menu can be used to set the color and attribute mapping for the pcwindow type terminals. This screen allows changing the mappings of the displayable attributes. For each of the character attribute combinations, the foreground color and intensity, and the background color and intensity be changed. Up to eight colors can be selected. If the color environment (ICCOLOR) has been set to process, these attribute-to-color defaults are NOT used. The actual character attribute is sent to the monitor unchanged.

When using a PCWINDOW or PCWINDOWCOLOR base, all 16 different attribute combinations can be specified. When using the PCWINDOWMONO base, only the 8 non-blinking attribute combinations are set, the blinking versions are set to match their non-blinking counterparts. Also when using PCWINDOWMONO setting the Normal set of attributes will copy them to the following selections where they can still be changed.

Note that the icrunw and the icrunrc programs support the bold and underline attributes via different fonts and blink is also supported.

Color Attribute Configuration

Character Attribute Combination	Foreground Color	Foreground Intense	Background Color	Background Intense
Normal	**Cyan**	**Y**	**Black**	**N**
Underlined	**Magenta**	**Y**	**Black**	**N**
Reversed	**Black**	**N**	**Cyan**	**N**
Reversed Underlined	**Black**	**N**	**Magenta**	**N**
Bright	**Green**	**Y**	**Black**	**N**
Bright Underlined	**Brown**	**Y**	**Black**	**N**
Bright Reversed	**Black**	**N**	**Green**	**N**
Bright Underlined Reversed	**Black**	**N**	**Brown**	**N**
Blink Normal	**Cyan**	**Y**	**Black**	**Y**
Blink Underlined	**Magenta**	**Y**	**Black**	**Y**
Blink Reversed	**Black**	**N**	**Cyan**	**Y**
Blink Reversed Underlined	**Black**	**N**	**Magenta**	**Y**
Blink Bright	**Green**	**Y**	**Black**	**Y**
Blink Bright Underlined	**Brown**	**Y**	**Black**	**Y**
Blink Bright Reversed	**Black**	**N**	**Green**	**Y**
Blink Bright Und Reversed	**Black**	**N**	**Brown**	**Y**

Press <up> or <down> to position,
<left> or <right> to change, ESC to exit.

SCREEN 20. COLOR ATTRIBUTE MAP

D.8. Change Directory

Option 7 from the TERMINAL DESCRIPTION menu can be used to change the default directory from which terminal description file are saved to or retrieved from.

D.9. Save and Retrieve Terminal Description File

Options 8 and 9 from the TERMINAL DESCRIPTION menu, allow for terminal description file(s) to be saved or retrieved with the `.tdi' extension. The Save(d) file and Retrieved file fields are used to prompt for the filename to be saved or retrieved respectively.

In general customized files should be saved to a unique directory that can be found by the runtimes with the ICCONFIGDIR environement entry.

E. Printer Translations (.pti)

E.1. Overview

Option 3 from the MAIN MENU will display SCREEN 21. This menu provides the ability to configure different printer translations files for use when printing through the @PRN's or @PCQ's. Printer translations are saved to files called printer translation files with a `.pti' extension. To build a base set of printer translation files, use the menu to select the appropriate base translation (selection 1) and then use save (selection 7) to create that default translation. The default translation should never be saved and used as that is just a one to one mapping of characters which will be done automatically if no printer translation is specified. On Linux, when using the Linux spooler, the Linux spooler should be configured to use the appropriate filter to perform the printer translation instead of using Interactive COBOL printer translation facility.

The name of the current base printer selection is displayed in this screen and the Configure Character Mapping screen.

```
          Printer Translation Configuration

    Name: default
    Base: default              Comment: Default

     1. Select Base Translation
     2. Change Comment
     3. Job Control String
     4. Configure Character Mapping

     5. Change Directory
     6. Save Printer Translation File
     7. Retrieve Printer Translation File

     Selection: 1

Directory:         [directory]
Retrieved file:
Save(d) file:

          Press <up>, <down>, to select, ESC to exit.
```

SCREEN 21. ICCONFIG PRINTER TRANSLATION (.pti)

To allow a particular customized printer translation to be available for Interactive COBOL, a printer translation file (.PTI) must be created for that particular entry. Each of the base printer translations can be used a a starting point. This can be done by using option 1 (Select Base Translation) followed by the needed change options followed by option 6 (Save) for each of the needed files.

All the base printer translation descriptions are available by default within the runtimes without needing the actual file. Only customized files need to be provided.

To change a printer translation file, use option 1 (Select Base Translation) or option 7 (Retrieve) to choose the translation to be changed. Now use option 4 (Configure Character Mapping) to change the needed values and then

use option 6 (Save) to store that printer translation. Option 2 (Change comment) can be used to store a comment about this new translation.

To build a new printer translation file, use option 1 (Select Base Translation) or option 6 (Retrieve) to choose a translation whose output characteristics most closely match the new translation. Now use option 4 (Configure Character Mapping) to change the needed values for the particular translation and then use option 6 (Save) with a new filename. These tailored versions should generally be saved with a different name and comment to distinguish them from the defaults. Option 2 (Change comment) can be used to store a comment about this new translation.

E.2. Select Base Translation

Option 1 from the PRINTER TRANSLATION menu positions to the Base Printer Translation field and allows a base translation to be selected from a list of available translations. These can be selected via a ferris wheel using the left-arrow and right-arrow keys. This base set includes the following:

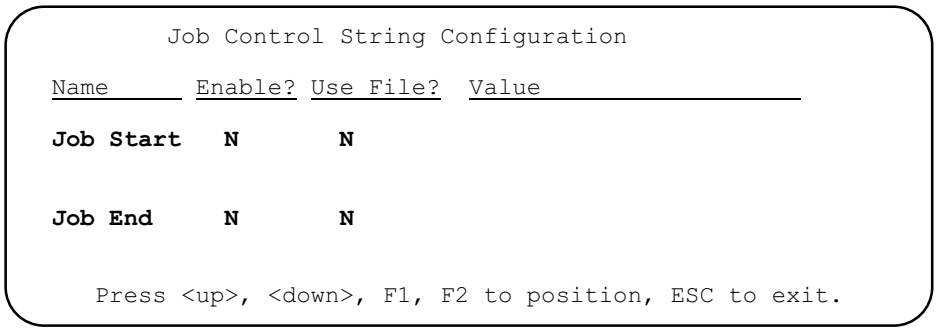
Default:	basic one to one mapping
DG to PC:	maps the DGI character set to the standard IBM PC character set
PC to DG	maps the standard IBM PC character set to the DGI character set

E.3. Change Comment

Option 2 from the PRINTER TRANSLATION menu can be used to insert a comment into a printer translation file. Selecting option 2 positions to the Comment field allowing up to thirty (30) characters to be entered for this printer translation. This option is especially useful if you change a base printer translation to provide some documentation of the change.

E.4. Job Control String

Option 3 from the PRINTER TRANSLATION menu can be used to insert job control strings at the beginning and/or end of a print file. Each can be separately set and can either be a series of values or a filename that contains the codes to be inserted.



SCREEN 22. PRINTER JOB CONTROL STRING CONFIGURATION

E.5. Configure Character Mapping

Option 4 from the PRINTER TRANSLATION menu presents SCREEN 23. The CHARACTER MAPPING CONFIGURATION instructs Interactive COBOL what to do for each possible output character from the COBOL program to the printer.

The current name of the printer being configured is displayed at the top of the menu.

This table provides entries for all 256 possible sequences with each line representing an entry as two major columns: Character from program and Character(s) to Print.

The first major column, Character from Program, gives the character from the program to be output to the printer in Decimal, Octal, Hex, and as a Description.

The second major column, Character(s) to Print, gives a Value column that allows entry of the value to be displayed along with an additional sub-column showing what this output character looks like on this terminal. Up to 11 bytes are allowed.

```
              Character Mapping Configuration
                      Name:    default
                      Base:    Default
Character from Program   Character(s) to Print
Dec Oct Hex Description  Byte String (Maximum 11 bytes)      Chr
--- --- --- -----------  ----------------------------------  ---
  0 000 00  Ctrl-@       \000                                N/D
  1 001 01  Ctrl-A       \001                                N/D
  2 002 02  Ctrl-B       \002                                N/D
  3 003 03  Ctrl-C       \003                                N/D
     .
     .
     .
254 376 FE  \376         \376                                ( )
255 377 FF  \377         \377                                ( )

Press <up>, <down>, F3, F4 to position, F5 to copy, ESC to exit.
```

SCREEN 23. CHARACTER MAPPING CONFIGURATION

When entering data in the Values column the following can be used:

a enters a bell (Ctrl-G)	*t* enters a tab (Ctrl-I)
b enters a backspace (Ctrl-H)	*v* enters a vertical tab (Ctrl-K)
dnnn enters nnn in decimal	*xnn* enters nn in hex
e enters an ESC	\\, \^, \″, \′, and *?* enters a single `\', `^', `"', `', and `?' respectively
f enters a form-feed (Ctrl-L)	^@ through ^_ enters the control code (\000 - \037)
n enters a linefeed (Ctrl-J)	*nnn* enters nnn in octal
r enters a carriage-return (Ctrl-M)	any printable character (`!' - `~') is entered as itself

When displayed in the Chr column, all non-control code characters (both 7- and 8-bit) will be displayed as defined within parenthesis ( ), while the control code characters will show a N/D (Not Displayable) in that column.

E.6. Change Directory

Option 5 from the PRINTER TRANSLATION menu can be used to change the default directory from which printer translation files are saved to or retrieved from.

E.7. Save and Retrieve Printer Translation File

Options 6 and 7 from the PRINTER TRANSLATION menu, allow for printer translation file(s) to be saved or retrieved with the `.pti' extension. The Save(d) file and Retrieved file fields are used to prompt for the filename to be saved or retrieved respectively.

In general customized files should be saved to a unique directory that can be found by the runtimes and ICEXEC with the ICCONFIGDIR environement entry.

F. Exit ICCONFIG

Pressing ESC while in the MAIN MENU terminates ICCONFIG and returns you to the program from which you called ICCONFIG.

G. Batch Update Facility (.cfi)

ICCONFIG supports a batch update facility. The -l|-L (Load .ini) switch is available to load a .ini file into the current (or new configuration). This facility uses standard .ini file processing to allow for non-interative changing of the configuration file.

The -l|-L file|dir (Load .ini), instructs ICCONFIG to use the .ini file specified to change the current configuration. Only those sections and line(s) that are being changed need to be present in the .ini file to be loaded. If a change is made, ICCONFIG will display that the file was updated. If no changes were made, no message will be displayed and the configuration file will not be updated. Only valid lines are read from the .ini file. The Load switch requires the Batch switch (-b).

The ICCONFIG .ini file provides the ability to specify an entire configuration file (.cfi). No support is provided for terminal description files (.tdi) or printer description files (.pti).

Rules for processing the .ini file:

1. Section names (names inside of [ ]) and keywords (names on the left of the =) are case sensitive.

2. Entries are not read from the .ini file in the order they appear in the .ini file. ICCONFIG reads sections and keywords within a section in a predetermined order (the order does match a generated file).

3. Invalid entries (either sections or keywords) are never read.

4. If more than one matching keyword in a section, only the first one is read.

5. Values for keywords are checked for correctness. An error is given with the keyword and section names. A special value <empty> is used/provided to denote no value for a keyword.

6. Except for trailing spaces on a line, spaces are significant.

7. Comment lines can be entered in the .ini file with the ";" character.

8. Valid section names are shown in a default .cfi file. By looking at a main section and looking at its default entries values you can see all the valid entries for that section.

So to perform a batch update, one would start with a valid .cfi file and pick the section(s) needed and copy them to the appropriate .ini file to be used for the batch update.

V. ICEXEC

A. Introduction

This chapter discusses ICEXEC, the global server daemon that allows the Interactive COBOL processes to share files and interact with one another.

ICEXEC is a daemon (global server) that coordinates execution of Interactive COBOL processes and configures the necessary global structures for shared memory and semaphores. Interactive COBOL requires one shared memory area and 4 semaphore sets. The ICEXEC utility must be running before any Interactive COBOL runtime (ICRUN, ICRUNCGI) or ICNETD server surrogate (ICIOS, ICRUNRS) is allowed to execute. All other Interactive COBOL executables (ICOBOL, ICCHECK, ICPACK, etc.) can be run with or without ICEXEC, as well as the ICNETD surrogate ICSQLS.

Note: Linux does not provide an exclusive open capability and so this is provided by ICEXEC. When ICEXEC is not running, exclusive open is emulated by posting a write-lock on the whole file. A non-exclusive open posts a read-lock on the whole file. Thus, two programs can detect whether a file is opened or open-exclusively by using this mechanism. Care should be exercised when moving from no-ICEXEC to ICEXEC-running, as utilities that started in the no-ICEXEC mode will keep running in that mode until they terminate.

B. Syntax and Startup

To execute ICEXEC you must be super user and the syntax is:

```
icexec [-a[:aflag]|-A file|dir[:aflag]] [-C file|dir] [-h|-?] [-i] [-O c|t]
   [-P file|dir] [-q] [-t][-T min[:max]]
```

Where

-a[:*aflag*]|-A *file*|*dir[:aflag]* (Audit)
: Enables auditing (default icexec.lg). Where *aflag* is a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-C *file*|*dir* (Configuration file)
: Specifies the configuration file. If only a directory is specified, the filename system.cfi is appended, otherwise the given file is used as the configuration file. If no Configuration file switch is given, system.cfi in the current directory is used.

-h|-? (Help)
: Display help text.

-i (Info)
: Put out Info messages.

-O c|t (Operation)
: Specifies either a Check(c) Operation or a Terminate(t) Operation to be performed. The check operation searches to see if another ICEXEC (of this major revision) is running. The terminate operation will terminate another ICEXEC (of this major revision) if it is running.

-P *file*|*dir* (Printer Queue location)
: If a directory is given (*dir*), it specifies that the default printer queuefile, system.pq is to be located in that directory. If a *file* is given, then the printer control file is then *file*.pq. If not specified, the default printer control file is system.pq in the current directory.

-q (Quiet)
: Enable quiet operation.

-t (Trace)
: Start with tracing amplified (turned on).

-T *min*[:*max*] (Terminal range)
: Specifies the minimum (*min)* and maximum (*max*) consoles to enable. If not specified, all enabled consoles are enabled.

Note: The Check Operation (-O c) does not require super user.

The basic install does configure the services and allows for the Linux service (ln6) or systemctl (ln7 & ln8) utilities to be used. See the install section for more information.

For a production system, ICEXEC should be started when the system moves from single user or maintenance mode to multi-user mode by whatever method is appropriate. ICEXEC should not be started via the inittab mechanism (ln6).

If ICEXEC is started on a pseudo-tty or on a tty that may log off from Linux such that ICEXEC can no longer log to the starting console, then the Quiet switch (-q) should be given when starting ICEXEC to prevent it from logging to the default console and possible hanging on a log write.

When ICEXEC starts it takes the following steps:

1) Processes and validates any command line switches.
2) The audit file is opened. If the audit file does not exist, it is created, and, if it does exist, it is opened with the erase option.
3) Checks that the user is in super user mode.
4) Checks and loads the configuration file.
5) Allocates a shared memory structure to be used by the Interactive COBOL process(s). The amount allocated is displayed.
6) Allocates and initializes four (4) semaphore sets.
7) Initializes shared memory and in the process it checks all Interactive COBOL logical device names to see if they exist. If a particular Linux device cannot be found, a warning message is generated.
8) Printer translation files (.pti) are opened and read when ICEXEC is started. If not found an error is given.
9) Checks and load the printer control file if needed.
10) ICEXEC then detaches from the current console and becomes a daemon.

If at any point in the above steps ICEXEC detects an error, it displays the appropriate message and terminates. If step 10 succeeds, the user on the invoking console will be back at the shell prompt and have received at least two (2) messages from the ICEXEC daemon that will show up as:

```
(date) (time) icexec (pid#):Info: Detached from login session context.
(date) (time) icexec (pid#):Info: The system is ready.
```

If, in step 5 or 6, an error is generated, you will most likely have to build a new kernel, changing the appropriate Linux system parameter to fix the problem.

Once ICEXEC becomes a daemon it sends any messages to the invoking console (unless the Quiet switch was given) and the audit file in the following format:

```
(date) (time) icexec (pid#):severity:text
```

Where
date is the current date,
time is the current time,
pid # will be the pid on which ICEXEC is running and is the pid # that must be used on a KILL to terminate ICEXEC,
severity shows the level of message severity, consisting of PANIC, ERROR, INFO, and WARNING, and
text message contains specific information.

If ICEXEC detects any error conditions while it is running, an appropriate message is sent to the audit file and the invoking console.

SCREEN 24 shows an example of the output when ICEXEC is invoked from a terminal. The first set of messages thru "**Detaching . .** ". come from the initial icexec process. The final messages starting with "**(*date*)(*time*) icexec (pid#): ...**) come from the daemonized icexec process.

```
[root@ln7x64vm ~]# icexec Revision 5.40 (Linux for x86 (ln7 64-bit))
Copyright (C) 1987-2020, Envyr Corporation.  All rights reserved.
Processing configuration file: /usr/cobol/system.cfi
Info: Shared memory area of 5946212 allocated.
Info: 50 MB of buffers allocated.
Info: 50 KB available in the process local buffer cache.
Info: Initializing shared memory area, please wait...
Processing printer control file: /usr/cobol/system.pq
Detaching from login session context.
 (date)(time)icexec (263):Info: Detached from login session conte
 (date)(time)icexec (263):Info: The system is ready.
[root@ln7x64vm ~]#
```

SCREEN 24. SUCCESSFUL ICEXEC STARTUP

If ICEXEC detects an expiration time after which the Interactive COBOL software will not be authorized to run, it will display a message to that effect. (This would be true for Beta releases.)

The Operation switch (-O) instructs ICEXEC to perform some operation. Valid operations are:

c (check)	Check for another ICEXEC with the same shared memory segment. This shared memory segment is usually the same for all minor revisions. If there is an ICEXEC running, the exit code is zero and the pid is returned. If no ICEXEC is running, an error is returned.
t (terminate)	Send a "kill SIGTERM" signal to another ICEXEC that is running with the same shared memory segment. You must be super user for terminate to work. If there is an ICEXEC running that can be terminated; the exit code is zero and the pid is returned. If no ICEXEC is running or it cannot be killed, an error is returned (non-zero exit code).

C. Termination

You must be super user to terminate ICEXEC. The recommended method to terminate ICEXEC is to use the appropriate system utilities.

For ln6 systems, the command is:
    # service icobol stopicexect
For ln7 & ln8 systems the command is:
    # systemctl stop icexec.service

However, ICEXEC, can also be terminated by using the Terminate Operation (-O t) of ICEXEC or by performing a KILL to the pid of the ICEXEC process. The default KILL signal (SIGTERM) is all that is necessary to shutdown ICEXEC and all Interactive COBOL processes connected to it. A SIGKILL (09) should never be used to terminate ICEXEC except as a last resort when the software and/or system is otherwise hung.

ICEXEC handles the following Linux signals with the given action:

SIGHUP (01)	- terminate ICEXEC
SIGINT (02)	- terminate ICEXEC if no Interactive COBOL processes are running
SIGQUIT(03)	- terminate ICEXEC
SIGTERM(15)	- terminate ICEXEC
SIGPWR (19)	- terminate ICEXEC

When ICEXEC terminates, it will first prevent any new Interactive COBOL processes from starting and then it will terminate all running Interactive COBOL processes by first issuing a SIGTERM to each process and, if that does not

stop the process, a SIGKILL (09) will be issued to the process. Thus, when ICEXEC terminates, all Interactive COBOL processes should be gone and no new ones can start.

A SIGKILL (09) should never be used to terminate ICEXEC except as a last resort when the software and/or system is otherwise hung.

In the example shown in SCREEN 24 above, the ICEXEC pid number is 263 so to terminate ICEXEC as super user you must do a:

```
kill 263
```

Please be patient while ICEXEC is terminating because it may take a significant amount of time to wait for various Interactive COBOL processes to terminate and to clean up various resources.

If ICEXEC is ever terminated abnormally, (e.g., if a SIGKILL (09) was used to terminate it), its shared memory area and semaphore sets are left allocated (i.e., not cleaned up properly). Either the Linux system must be rebooted, the Linux *ipcrm* utility can be used to clean up these resources, or ICEXEC can be restarted in which case it will clean up the resources properly and shutdown. You then must restart ICEXEC again to actually bring ICEXEC up.

On a system shutdown, the system shutdown script/executable will send the appropriate signals to ICEXEC to cause it to start shutting down.

When ICEXEC terminates it will list any open files and whether they have been modified. It will also dump a usage table that can be used to help tune the **ICOBOL** configuration.

If ICNETD I/O servers (icios) are used while ICEXEC is running, then logical devices (@PCQ.., @SER.., @PRN...) may be used by remote clients.

D. Processing

ICEXEC maintains the global shared area to which all **ICOBOL** processes that deal with files attach. It continuously monitors active processes. If ICEXEC detects that a process terminates unexpectedly for any reason it checks to see if that process had any files opened and if they had been modified. If so, ICEXEC will open the affected file(s) and flush any modified buffers to the disk. If they had not been modified, ICEXEC just clears the shared area information for that file. ICEXEC will log any unexpected error conditions that it detects along with what it has to do to keep the files and shared area in a consistent state. Generally unexpected terminations should be investigated to determine their cause and especially if process traps or core dumps the problems should be reported.

On Linux, ICEXEC also communicates with *lpsched* to keep the printer control system up-to-date.

VI. STARTING ICRUN

A. Introduction

This chapter discusses how to execute the Interactive COBOL runtime. These COBOL program files (.CX) can be produced from COBOL source files with the Interactive COBOL compiler (ICOBOL).

In this chapter, Interactive COBOL will be used to refer to all versions of Interactive COBOL on Linux.

B. Environment Entries

B.1. Overview

ICRUN searches for specific Interactive COBOL entries in the user's current environment. These entries allow the user to tailor a particular session of Interactive COBOL for a particular terminal type, user, application, company, etc.

If an entry is found in the environment, it overrides any similar entry in the configuration file.

Interactive COBOL environment entries other than ICROOT, ICCONFIGDIR. ICTMPDIR, and ICPERMIT_MACHINE that are used are:

DATAFILE	Generic data file (@DATA)
ICABORT	Enable aborting on global timeout
ICBGCOLOR	Specify the initial background color
ICCODEPATH	PATH for .CX files
ICCOLOR	Specify how to support color
ICCOLUMNS	Columns for terminal
ICDATAPATH	PATH for data files
ICFGCOLOR	Specify the initial foreground color
ICLINES	Lines for terminal
ICNETUSESHEARTBEAT	Specifes whether to connect to the icios surrogate with a heartbeat
ICPCQDIR	Directory for spooled print files
ICPCQFILTER	Specify a default Printer Control Utility Filter
ICPROMPTCHAR	Specify the prompt character
ICREVERSE	Specify how to support reverse
ICRUN	Default switches for the runtime
ICRUNDIR	Default directory
ICRUNLK	Link file
ICSCROPT	SCREEN OPTIMIZER selection
ICSDMODE	SCREEN HANDLER selection
ICTERM	Terminal type
ICTIMEOUT	Specify global timeout for ACCEPT
LISTFILE	Generic list file (@LIST)
PCQ	Generic printer control queue
PRN	Generic printer device
PTS	Generic device for printer-pass thru
SER	Generic serial device

When using the Bourne shell these entries must be exported to be made available to ICRUN.

B.2. DATAFILE

DATAFILE specifies a filename for the @DATA file.

The syntax is:

```
DATAFILE=filename
```

Where
filename is a filename to be used to replace @DATA in a COBOL OPEN.

B.3. ICABORT

ICABORT specifies whether to abort the Interactive COBOL process if a global timeout (ICTIMEOUT) occurs.

The syntax is:

```
ICABORT=on|off
```

If not specified in the environment then the ICABORT from the configuration file is used.

If ICTIMEOUT is not set or has been overridden by an ACCEPT with TIME-OUT statement or the IC_SET_TIMEOUT builtin, the abort is ignored.

B.4. ICBGCOLOR, ICCOLOR, ICFGCOLOR

ICCOLOR instructs Interactive COBOL how to interpret color codes from a COBOL program. Valid selections are filter, ignore, and process; the default is filter.

The syntax is:

```
ICCOLOR=filter|ignore|process
```

Where
filter
Causes the runtime to watch for color codes from the program and to NOT send them to the terminal, since it does not support color. Filter is the default.
ignore
Tells the runtime that the user wants total control of the screen and may be sending binary color data to the screen and that the runtime should ignore all color codes (i.e., do not look for color codes). If running in this mode, the SCREEN OPTIMIZER should not be enabled as the runtime cannot accurately repaint a user's screen.
process
The runtime interprets color codes from the program and sends the appropriate sequences to the terminal. When set to Process the initial background and foreground colors are set by the runtime at startup.

ICBGCOLOR sets the default background color to the indicated value when running with ICCOLOR set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7) either as the name or the number. The default is black (0).

The syntax is:

```
ICBGCOLOR=black|blue|green|cyan|red|magenta|brown|white|
            0|1|2|3|4|5|6|7
```

ICFGCOLOR sets the default foreground color to the indicated value when running with ICCOLOR set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7) either as the name or the number. The default is white (7).

The syntax is:

```
ICFGCOLOR=black|blue|green|cyan|red|magenta|brown|white|
            0|1|2|3|4|5|6|7
```

If not specified in the environment then the appropriate entry from the configuration file is used.

Currently only DG and Ansi based terminals support color.

B.5. ICCODEPATH

ICCODEPATH specifies directories or COBOL library files (up to 16) that ICRUN will search to find a COBOL program (.CX) file when a simple program name is encountered in a CALL, CALL PROGRAM, or from the command line..

The syntax is:

```
ICCODEPATH=dir|file[:dir|file]...
```

Where

dir
: Specifies a directory in which ICRUN should look for the .CX files of programs with simple names.

file
: Specifies a COBOL library file in which ICRUN should look for the .CX files of programs with simple names. COBOL Library files are built with the ICLIB utility.

If not specified in the environment, then the ICCODEPATH entry from the configuration file is used. If neither is specified, a default ICCODEPATH of the current directory is used.

If ICCODEPATH is specified, the current directory is searched last by default and must be specified by an empty entry or a period in ICCODEPATH if you wish it to be searched sooner. The list is searched for programs (.CX files) in the order given in ICCODEPATH.

ICCODEPATH must only include directory names or COBOL library names.

The ICCODEPATH only applies to a program with a simple name. The syntax is the same as for a system PATH.

A sample entry is:

```
ICCODEPATH=.:/usr/bin/pgms:/usr/pgms1:/usr/master/icobol.cl
```

which searches the current directory, /usr/bin/pgms, /usr/pgms1, and finally the COBOL Library file icobol.cl in the /usr/master directory for a particular .CX file.

ICCODEPATH is processd at startup.

B.6. ICTERM, ICCOLUMNS, ICLINES

ICTERM tells Interactive COBOL what type of terminal is attached to a particular line. ICLINES and ICCOLUMNS specify the number of lines and columns for the terminal.

The syntax is:

```
ICTERM=terminal-type
ICLINES=lines
ICCOLUMNS=columns[:ccolumns]
```

Where

terminal-type
: Specifies a valid ICTERM entries described below or in the ICTERM Chapter starting on page 125, with a corresponding terminal description file

lines
: Specifies the number of lines for this terminal. It can range from 24 to 255.

columns[:*ccolumns*]
: Specifies the number of columns for this terminal. It can range from 80 to 255. If the second value is given, then both normal and compressed mode are supported and the larger value is the compressed number of columns. The first value given is the mode that the terminal starts in.

If not specified in the environment then the ICTERM from the configuration file is used.

If ICLINES and/or ICCOLUMNS are not specified, or are 0, then that specified in the configuration file is used if non-zero. If ICLINES and/or ICCOLUMNS are zero in the configuration file, then the numbers in the appropriate terminal description file are used except for the terminfo and pcwindow terminal descriptions. For terminfo descriptions, ICLINES and/or ICCOLUMNS are those defined by the LINES and/or COLUMNS from the terminfo database. For pcwindow descriptions, ICLINES and/or ICCOLUMNS are those defined by the video bios. If no ICLINES or ICCOLUMNS information can be found, the defaults of 24 lines by 80 columns are used.

Current valid default ICTERM terminal-types (all lower-case) and a brief description are given below:

aix	the high function console (hft) on an AIX machine,
ansi	a standard ansi terminal (like a vt100 but no function keys),
att	an AT&T 605 type terminal,
dg	a DG D200 or above type terminal,
dgunix	a DG D217+ terminal in dgunix mode
freedom	a Freedom ONE type terminal,
terminfo	any terminal defined in the terminfo database,
ibm	an IBM 3101 or 3151 type terminal,
linux	a Linux master console,
mac	a Terminal session on Mac OS X,
sun	a SUN master console,
vt100	a VT100 type terminal,
vt220	a VT220 type terminal,
wyse	a Wyse 50-plus type terminal,
wy50	a Wyse 50 type terminal with NO attribute support,
sco, xenix	an SCO master console or SCO master console type terminal,
xterm-fk	an XWINDOW terminal emulator
386ix, at386	a 386/ix or AT&T 386 master console

More on these terminal types can be found in the ICTERM Chapter, starting on page 125.

When using CGI, ICTERM=file should be set.

ICLINES should be set to the line at which the terminal will scroll the screen when a line-feed (<lf>) is sent. (Line-feed and newline are the same.)

ICCOLUMNS should be set to the column position after which the terminal will wrap to the next line. If both normal and compressed spacing is available then two values should be specified.

If a terminal is supported directly by Interactive COBOL, it should be used instead of using terminfo because Interactive COBOL generally provides better performance and memory usage.

B.7. ICDATAPATH

ICDATAPATH specifies directories and/or COBOL libraries (up to 16) that ICRUN will search to open a COBOL data file with a simple name.

The syntax is:

```
ICDATAPATH=dir[:dir]...
```

Where

dir

Specifies a directory in which ICRUN should look for data files with simple names.

If not specified in the environment, then the ICDATAPATH entry from the configuration file is used. If neither is specified, a default ICDATAPATH of the current directory is used.

If ICDATAPATH is specified, the current directory is searched last by default and must be specified by an empty entry or a period in ICDATAPATH if you want it to be searched sooner. The list is searched for data files in the order given in ICDATAPATH.

ICDATAPATH must only include directory names.

The ICDATAPATH only applies to a data file with a simple name. The syntax is the same as for a system PATH.

A sample entry is:

```
ICDATAPATH=.:/usr/bin/data:/usr/data1:/usr/master
```

which searches the current directory, /usr/bin/data, /usr/data1, and finally the directory /usr/master for a particular data file.

ICDATAPATH is processed at startup.

B.8. ICNETUSESHEARTBEAT

ICNETUSESHEATBEAT specifies whether a client/server connection using the ICNET i/o surrogate icios would be opened with a headrtbeat connection. When opened with a heart beat connection the two processes periodically send “heartbeat” packets to ensure that a connection is active. Upon the loss of the connection the icios surrogate will cleanly terminate. This will happen after two heartbeats are lost, usually within 120 seconds

The syntax is:

```
ICNETUSESHEARTBEAT=1
```

ICNETUSESHEARTBEAT is processed on the first client/server connection. I.E. an open using the following type of pathname, "@[icnet:]\\machine-name[:port]\path".

ICNETUSESHEARTBEAT requires 4.40 or above on both the client and server.

B.9. ICPCQDIR

ICPCQDIR specifies a particular directory in which simple printer filenames should be created (printer control directory) when Printer Control is enabled. ICPCQDIR is not used otherwise.

The syntax is:

```
ICPCQDIR=dir
```

Where

dir

Specifies a valid pathname for the directory in which COBOL printer files with simple names are to be located.

If not specified in the environment then the ICPCQDIR from the configuration file is used. If neither is specified, printer files are located in the current directory.

The Printer files affected are those with an ASSIGN TO PRINTER filename and ASSIGN TO PRINTER-1 filename where the filename specified is a simple filename, i.e., no directory qualifiers. Each user can have his own printer control directory or there can be a common one for a group of people or the entire machine.

B.10. ICPCQFILTER

ICPCQFILTER specifies a particular printer control utility filter. The display may be modified for the life of the runtime by setting the ICPCQFILTER environment variable to establish the default filter. The specified filter will be active every time the Printer Control Utility is entered. This method applies to all methods of starting the Printer Control Utility (IC_PRINT_STAT).

ICPCQFILTER, takes values in a format similar to the command lines of the Interactive COBOL utilities.

The syntax is:

```
ICPCQFILTER=command-line
```

Where *command-line* includes:

-B *min*:*max*

Printed by user-id numbers ranging from *min* to *max*

-D *directory*

Pathname of directory directly or indirectly containing the print job

-F *filename*

Simple filename of the print job

-I *min*:*max*

Owned by user-id numbers ranging from *min* to *max*

-M *mode*

The job's current status. Valid modes are:

1=Not yet printed	2=Already printed	3=Error occurred	4=Update in progress
5=Queued to print	6=Holding in queue	7=Printing	8=Retrying
9=Terminating			

-O *owner*

Username of the owner

-P *printedby*

Username of the last user to print the file

-Q *min*:*max*
Printer control queues ranging from *min* to *max*
-r
Read access to a file
-S *min*:*max*
Filesize (in bytes) ranging from *min* to *max*

None of the command line options may be specified more than once. Each must be separated by a space.

For example, if ICPCQFILTER is set to "-O mary -Q 2:3", the Printer Control Utility would only display files owned by the user "mary" and destined for either @PCQ2 or @PCQ3.

B.11. ICPROMPTCHAR

ICPROMPTCHAR specifies an alternate prompt character for screen input fields. The default character is the underscore (_).

The syntax is:

```
ICPROMPTCHAR=x
```
where x is a single display-able character

B.12. ICREVERSE

ICREVERSE informs Interactive COBOL how to interpret reverse codes from a COBOL program.

The syntax is:

```
ICREVERSE=filter|ignore|process
```

Where
filter
Causes the runtime to watch for reverse codes from the program and to NOT send them to the terminal, since it does not support reverse.
ignore
Tells the runtime that the user wants total control of the screen and may be sending binary reverse data to the screen and that the runtime should ignore all reverse codes (i.e., do not look for reverse codes). If running in this mode, the SCREEN OPTIMIZER cannot correctly repaint a user's screen that includes reverse.
process
The runtime interprets reverse codes from the program and sends the appropriate sequences to the terminal. Process is the default.

If not specified in the environment then the ICREVERSE setting from the configuration file is used.

B.13. ICRUN

The contents of the *ICRUN* environment variable are treated like switches entered from the command line and processed before any other switches or arguments when starting ICRUN.

The syntax is:

```
ICRUN=icrun-switches
```

Where
icrun-switches

Specifies any valid command line switches for ICRUN but not a command-line argument.

B.14. ICRUNDIR

ICRUNDIR specifies a particular directory that the runtime will position to after it has successfully processed the command line and optionally opened its audit log. All subsequent current directory usage will use this value

The syntax is:

```
ICRUNDIR=dir
```

Where

dir

Specifies a valid pathname for the directory into which the runtime will position to just before it starts to run the initial program.

If the specified value is NOT a valid directory, an error will be given at startup and the runtime will exit.

If the Info switch (-i) is given, a message will be given about processing the ICRUNDIR environment variable.

ICRUNDIR is processed before ICDATAPATH and ICCODEPATH.

B.15. ICRUNLK

ICRUNLK specifies where the link file can be found.

The syntax is:

```
ICRUNLK=file
```

Where

file

Specifies a valid filename for the link file.

If not specified in the environment then the ICRUNLK from the configuration file is used. If neither is specified, ICRUN does not look for a link file.

If specified in one form or other, then the link file must exist, be readable, and be a valid link file.

The entry can be a directory (in which case icrun.lk is appended for the file) or a complete filename. This feature can be used to link RDOS, MS-DOS, or AOS/VS type filenames to Linux type filenames without changing the COBOL programs. This file is created with the ICLINK utility explained in the Utilities Manual. The link file is opened, read, and closed at startup.

B.16. ICSCROPT

ICSCROPT informs Interactive COBOL and ICCONFIG whether to enable the Interactive COBOL SCREEN OPTIMIZER.

The syntax is:

```
ICSCROPT=off|on|full|partial|mute
```

Where

off
: Disables screen optimization

partial
: Enables simple single screen optimization

on and full
: Enables full screen optimization

mute
: Disables any screen optimization and prevents Interactive COBOL from sending any implied codes of its own at startup or termination.

If not specified in the environment then the ICSCROPT entry from the configuration file is used.

The SCREEN OPTIMIZER keeps track of all data sent to the console and prevents rewriting the same data multiple times. It uses an image of the current screen always in memory.

Usually screen optimization will provide improved screen performance.

The full option usually provides better screen update performance than the partial option, but it requires more memory and cpu time.

B.17. ICSDMODE

ICSDMODE instructs Interactive COBOL how to enable the Interactive COBOL SCREEN HANDLER that provides many SCREEN DEMON™ like features.

The syntax is:

```
ICSDMODE=disabled|underline|0|reverse|1|linedraw|2
```

Where

disabled
: Disables the SCREEN HANDLER

0 or underline
: Run in standard SCREEN DEMON format, which is to underline the row above the box and underline the last row in the box for the top and bottom lines, and use reverse video for the sides.

1 or reverse
: Use reverse video for the entire box. This means that two (2) more lines than in standard mode are hidden under the box.

2 or linedraw
: Use the line drawing character set of a terminal for the entire box. As in 1 above, two (2) more lines than in standard mode are hidden under the box. If a particular terminal does not have a line drawing character set +, -, and | are used for the corners, horizontal, and vertical portions of the box, respectively. Currently, only the terminal types ibm, xenix, 386ix, and pcwindow support the line drawing characters by default.

If not specified in the environment then the ICSDMODE from the configuration file is used.

For more information on the Interactive COBOL SCREEN HANDLER, see the Screen Handler section in the ICRUN Chapter of the Interactive COBOL Language Reference & Developer's Guide.

B.18. ICTIMEOUT

ICTIMEOUT specifies a global timeout in seconds for all screen ACCEPTs (and STOP literal). If no key has been hit in that time, the ACCEPT will timeout with the ESCAPE KEY set to 99.

The syntax is:

```
ICTIMEOUT=n
```

Where

n

Valid timeout values are:

<=0 or >=65535	Wait Forever
>6300 and <65535	Set to 6300
1-6300	Set to *n* seconds

6300 seconds is 1 hr. & 45 min.

If not specified in the environment, then the ICTIMEOUT from the configuration file is used.

The ACCEPT with TIME-OUT statement and the IC_SET_TIMEOUT builtin will override this selection.

B.19. LISTFILE

LISTFILE specifies a filename for the @LIST file.

The syntax is:

```
LISTFILE=filename
```

Where

filename is a filename to be used to replace @LIST in a COBOL OPEN.

B.20. PCQ, PRN, and SER

PCQ, PRN, and *SER* allow the generic printer control queue (@PCQ), the generic printer (@PRN), and the generic serial (@SER) devices to be specified for each process.

The syntax for each is:

```
PCQ=n
PRN=n
SER=n
```

Where

n

Specifies a number from 0 to 2047 for PCQ, PRN, and SER that is the appropriate device (@PCQn, @PRNn, or @SERn) enabled and defined in ICCONFIG.

If not specified in the environment, then that specified in the configuration file is used.

B.21. PTS

PTS allows a specific device or output-pipe to be set for printer-pass thru in this session. If set, this interception is done BEFORE the data is sent to the particular terminal.

The syntax is:

PTS=*device* | *output-pipe*

Where

output-pipe

is an output pipe with a leading bar greater than sign as: "|>lp"

For example:

```
PTS=/dev/lp
export PTS
```

or

```
PTS="|>lp"
export PTS
```

The first would cause the device /dev/lp to be opened and all data sent to that device.

The second will pipe the output to the lp command using the default printer.

Note that the bar and greater than sign must be escaped to prevent the shell from processing them.

C. Syntax

ICRUN is the program that "runs" a COBOL program. To run the runtime system, the ICEXEC program must be running to provide a shared memory area for communication and an ICPERMIT program must be available to provide licensing services.

Linux is a case-sensitive operating system as opposed to any DG proprietary operating system (where **ICOBOL** originated) or Windows. By default, the Interactive COBOL runtime converts all filenames (including programs) to lower-case before passing them to Linux. To open an upper-case or mixed-case file, you can create a Linux link (either a hard link or symbolic link) to that name using only lower-case letters or you can start the Interactive COBOL runtime with the appropriate Case switch (-C) to allow the filenames desired.

The syntax for the Interactive COBOL runtime is:

```
icrun [-a[:aflag]|-A file|dir[:aflag]] [-b] [-C l|n|u]
    [-D yyyymmdd[:hhmmss]] [-E var=value][-G {drsu}] [-h|-?] [-i] [-I 2]
    [-N {beinow}] [-p] [-q] [-S os] [-T n] [-U l|n|u] [-z|-Z ddir]
    [[[-s] program] [argument]...]
```

Where

-a[:*aflag*]|-A *file*|*dir[:aflag]* (Audit)

Enables auditing (default icrun.lg). Where *aflag* is a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-b (Batch mode)

Use for icrun with pipes or redirected i/o

-C l|n|u (Case)

Specifies the default case for all COBOL filenames. Valid cases are lower-case (l), no conversion (n), or upper-case (u). The default is lower-case (l).

-D yyyymmdd[:hhmmss] (Date bias)

Specifies a date with an optional time from which to bias all COBOL date and time functions (ACCEPT FROM DAY, DATE, TIME, DAY-OF-WEEK, the IC_FULL_DATE builtin, message sending, etc.). The current date/time is subtracted from this date/time and the resultant value is added to all date and time functions. No date/time changes are made to the operating system. This can be used to set a date/time forward or backward for testing purposes.

-E *var*=*value* (set Environment value)

Specifies that the environment variable *var* should be setup with the given *value*. This is especially useful for ThinClients to pass environment information over to the server runtime..

-G {drsu} (General)

Specifies a general switch value. Valid general switch values are:

d (Duplicates)	Causes the runtime to generate a FILE STATUS 02 with ANSI COBOL 74 programs when using ICISAM version 7 files.
r (RDOS)	Uses the first underscore in an ACCEPT field as a terminator that erases all characters to the right of it from the string of characters entered into the field. This behavior is how Interactive COBOL on RDOS operated.
s (Switch)	Enable strict switch processing. The first slash in a program name is treated as the beginning of the switches.
u (Upper-case)	Causes the runtime to upper-case program names returned by ACCEPT FROM ENVIRONMENT.

-h|-? (Help)

Display help text.

-i (Information)

Displays all Information (Info) messages while starting. The default is to not display information messages.

-I 2 (**ICOBOL** 2)

Run **ICOBOL** 2 .cx files. Ths is no longer required as the runtime will auto detect an **ICOBOL** 2 .cx

-N {beinow}... (No options)

Specifies NO option(s). Valid option values are:

b (No-reassignment)	do not allow as a reassigned job. I.E., this job must start connected to a configured @CONn logical device
e (No-embedded space)	do not allow embedded spaces in filenames
i (No-interrupts)	do not allow console interrupts
n (No Nagle suppression)	do not suppress the Nagle algorithm (may decrease network traffic but also typically reduces interactive responsiveness)
o (No-OCCURS)	do not do OCCURS DEPENDING ON bounds checking
w (No-warnings)	do not display Interactive COBOL warning messages at startup

-p (Prompt for username)

Always Prompt for username to pass to the server on a remote connection (ICNETD).

-q (Quiet)

Enable quiet operation.

-s (Startup)

Run *program* in startup mode. Uses the specified program only as the initial startup-program and then runs in Logon mode.

-S *os* (System-code)

Overrides the default system code returned by an ACCEPT FROM ENVIRONMENT. *os* must be a value from 00 to 99. This may be needed for older applications that check the OS code and do not account for the new OS codes for 64-bit systems.

-T *n* (Terminal number)

Sets the console number returned by Interactive COBOL, i.e., an ACCEPT FROM LINE will return this value. To use this switch the indicated console (@CONn) must have no device name specified and be enabled in the CONSOLE Configuration under ICCONFIG, and must not already be in use.

-U l|n|u (Username)

Specifies the case in which user name will be returned in the ACCEPT FROM USER NAME statement. Valid cases are upper-case (u), lower-case (l), or no conversion (n). The default is no conversion (n).

-z | -Z *ddir* (Debug)

Instructs the runtime to start in debug mode using the current directory (-z) or the directory *ddir*) as the default directory for symbol files (.sy).

program

Specifies a COBOL program name including optional program-switches. If program is given with no Startup-program switch (-s), Interactive COBOL will run in Program mode in which any program termination will cause Interactive COBOL to terminate.

argument

Provide optional arguments that will be passed to the specified program's linkage section. This works just like a CALL PROGRAM program USING arguments, . . . statement. The arguments are moved to the linkage section as if a COBOL MOVE had been done. If more arguments are given than are in the linkage section, the remaining arguments are discarded. If fewer arguments are given then no data is moved into any remaining linkage items.

The runtime starts in one of two modes. If no COBOL program is specified, it starts in Logon mode using a default program called logon.cx. If a COBOL program is specified and no -s switch is specified, the runtime starts in Program mode. Logon mode and Program mode respond to program termination in different ways.

When using the Prompt for username switch, the Save option is ignored in the username/password screen. For thick clients, the prompt is done when a connection is made to a network resource via ICNETD.

More on program switch processing can be found in the CALL PROGRAM statement in the Language Reference Manual.

When using an audit log with the runtime:

A) If any Error is encountered (like Indexed out of Range, etc.) the message will be written to the audit log along with the pc where the error occurred, the next pc, the exception status register, and the name of the program.

B) If an asynchronous event (like a Ctrl-C, Ctrl-Break, a Windows-Close, an Abort or Kill from another console) is encountered, then that message will be written to the audit log along with the pc where the error occurred, the next pc, the exception status register, and the name of the program.

The format for the above cases:

```
Error:  message
    OP=nnn1 PC=nnn2 E=nn3 in program-name
```

Where

nnn1	is the current pc
nnn2	is the next pc that would have been executed
nn3	is the current exception status
program-name	is the current COBOL program.

When running in logon mode, there could be multiple of these type of messages in the audit log as the user would only need to hit newline to continue from these errors back to the startup program.

Running with an audit log is always recommended, as it will allow you to look back over events if needed. In general, these runtime audit logs should be small. The Audit option with pid is probably the best type of auditing to use.

```
icrun -a:p
```
(for example)

Then you can use the pid number to look up the process in the icpermit and/or icexec logs if needed.

A common usage for Program mode would be:

```
icrun mainjob
```

which would run the program mainjob.cx in Program mode and on a STOP RUN immediately terminates and returns to the executing process.

The No-reassignment switch (-N b) and the Terminal number switch (-T) can be used together to prevent a particular Interactive COBOL process from starting unless it can control the specified @CON device.

The Interactive COBOL executable file must be found along the user's current PATH. The ICEXEC program must be running to allow Interactive COBOL runtime execution.

ICRUN searches the user's environment for all the entries in the previous section.

To start the runtime system from the command line type "icrun" and press the ENTER key. You should now see the following messages.

For a startup with no problems:

1) The Interactive COBOL revision information and copyright notice are displayed if the Quiet switch (-q) was not given.
2) The initial program is loaded and executed.

For a startup with problems, below is the general order of checks that Interactive COBOL makes at startup and the type of error message that can be generated, followed by (in parenthesis) what can be done to remedy the problem.

1) The ICRUN revision information and copyright notice is displayed if the Quiet switch (-q) was not given. If ICRUN cannot be found along the current PATH or cannot be loaded for any reason a system error is displayed instead. (Fix the PATH or change the needed permissions to load Interactive COBOL.)
2) ICRUN connects to the license manager(ICPERMIT). If no ICPERMIT is running, the following will be generated:
   ERROR: Connection refused ... Connecting to localhost:7334
   ERROR: The required software license is not authorized.
   (ICPERMIT must be started.)
3) ICRUN uses the global shared memory area setup by the ICEXEC program. If the area is not found then a message that "The global control area could not be accessed" is displayed. (ICEXEC is probably not running, please start ICEXEC.)
4) As it initializes, ICRUN allocates additional memory as required and if it runs out of memory, an error like "Failure allocating xxx" will be displayed.
5) The ICRUNLK (link file) entry is processed and validated. If it is an invalid entry or the file cannot be found or read, the appropriate error is displayed. (Either delete the ICRUNLK entry or fix the link file.)
6) The ICCODEPATH and ICDATAPATH entries are processed and validated. If either has an invalid entry the appropriate error is displayed. (Either delete the ICCODEPATH or ICDATAPATH entry or fix the path.)
7) ICRUN checks with ICEXEC to see if this terminal has been mapped to an enabled Interactive COBOL console (@CONn) that can run a program. If not mapped to an enabled console, this process is automatically reassigned to a console with a blank device name field, (unless the No-reassignment switch (-N b) was specified). If mapped to an enabled console, it then checks to see if Interactive COBOL is already running on this console; if so, a warning message is generated that the console is already in use and this process is then automatically reassigned to a console with a blank device name field (unless the No-reassignment switch (-N b) was specified). This allows multiple Interactive COBOL processes to be running on the same terminal but provides unique console numbers to each process. If a console is enabled, but not authorized to run programs, Interactive COBOL will give an error and terminate.
8) The ICTERM (terminal type) entry along with ICCOLUMNS, ICLINES, ICSCROPT, and TERM (if needed) are processed and validated. If any are invalid, the appropriate error is displayed. (Select a valid terminal type or fix the appropriate variable.)
9) The ICSDMODE, ICSCROPT, ICCOLOR, ICBGCOLOR, ICFGCOLOR, ICREVERSE, ICABORT, and ICTIMEOUT entries are processed and validated. If an invalid entry, the appropriate error is displayed.
10) The maximum number of allowable runtime users is checked. If at the maximum, the appropriate message is displayed. (Wait till someone logs off an Interactive COBOL process or upgrade to a larger license.)
11) Finally, the initial program is sought and loaded. If it cannot be found or it cannot be loaded for any reason, the appropriate error is displayed. (If the program cannot be found, check the COBOL library or ICCODEPATH.)

In each of the above error cases ICRUN terminates.

D. Termination

ICRUN terminates under program control through the IC_SHUTDOWN or IC_HANGUP builtins. Interactive COBOL can be terminated under user control if console interrupts are allowed. The standard Linux Intr and Quit interrupts are processed. Interactive COBOL can also be terminated by another process using the appropriate *kill* command (and the process has the proper privileges). Under normal circumstances, a *kill -9* should never be used to terminate any **ICOBOL** process (including ICRUN, ICEXEC, ICPERMIT, etc.).

To terminate the complete Interactive COBOL system (all Interactive COBOL processes) and prevent anyone else from starting ICRUN, the ICEXEC program must be terminated.

VII. ICNETD

A. Introduction

ICNETD provides the TCP/IP network communication and security handling for the server side of the client/server operations. These include the file i/o server (icios), the ThinClient server (icrunrs), and the Remote ISQL server (icsqls) for **ICOBOL**. Clients that use the i/o handler (icios) will generally be referred to as "ThickClients".

A TCP/IP network must be running between the client and server machines with the necessary daemons and drivers. The ping utility can be used from both the client and the server to check that each can access the other with the given machine name or IP address.

ICNETD I/O client/server (icios) is required in order to share files between Linux machines and Windows machines and between two or more Linux machines. Sharing files via NFS mounted file systems is NOT supported or recommended! The ICNETD I/O client/server (icios) often enhances performance when sharing files between two or more Windows machines vs. sharing files via a shared network drive..

The ICNETD I/O client access can be used over the Internet to access remote files. If a firewall is being used on the server machine, the port used by ICNETD (default 7333) must be opened.

I/O Client (thickclient)

I/O Client support is offered by the Linux and Windows runtime systems, the ODBC driver (ICODBCDR), and the user library. This is generally referred to as thickclient mode. The I/O client/server model differs from the traditional **ICOBOL** support for remote file access in that it acts at the COBOL operation level rather than the operating system operation level. In other words, it remotely reads and writes records rather than disk blocks. For complex files, like indexed files, this generally provides enhanced I/O performance in the network environment while reducing network traffic.

The I/O client (icios) requires a separate **ICOBOL** Network Server License in the license description file that ICPERMIT manages in order for the server to service clients in i/o client mode (no runtime licenses are used in this mode). Each simultaneous connection (icios process) requires a user count in the license.

ThinClient

ThinClient client (icrunrc) support is offered on all machines and gui support (sp2/qpr) is available under Windows. In the thinclient cases, only a small part of the code is on the client machine. Just enough to display the provided information and provide keyboard input support.

When providing ThinClient (icrunrc) support, both an **ICOBOL** Runtime license and an **ICOBOL** Network Services license must be available to start the ThinClient server (icrunrs). In addition, if any gui is to be used (sp2 or qpr) then an **ICOBOL** SP2RUN license will be required. ThinClient is similar to telnet support but is done all in **ICOBOL** space via an encrypted interface with the additional support for the gui components of sp2 and qpr and an automatic reconnection ability.

When using ThinClient client (icrunrc) and connecting to a Linux machine (thru ICNETD), the basic environment variables normally set by Linux logon are set by ICNETD. These are: HOME, LOGNAME, MAIL, and SHELL. The MAIL entry is set only if ICNETD is given the default MAIL path by using the ICNETD_MAIL environment variable to provide the path to the standard mail directory, to which ICNETD will add the username. For example, if ICNETD_MAIL is given as /usr/spool/mail/ then when the user "joe" logs on via a ThinClient, then the MAIL environment variable will be set to "/usr/spool/mail/joe".

The SHELL entry will only be set if the shell value is provided by the passwd file.

ISQL Client

An ISQL client is similar to the I/O Client, but it works at the SQL request level instead of the file level. It is used to support runtime system applications that use the **ICOBOL** imbedded SQL (ISQL) feature.

When providing ISQL support, an **ICOBOL** Network Server license must be available to start the ISQL server (icsqls). The ISQL server communicates with the ODBC Administrator and the ODBC Driver manager only on the remote machine. If the ODBC driver being used is the ICOBOL ODBC driver, then an ISQL connection will also require an ICODBCDR license for each copy of icsqls that is running.

B. Syntax

The standard syntax is:

```
icnetd [-a[:aflag]|-A file|dir[:aflag]] [-d] [-h|-?] [-L file|dir[:aflag]]
   [-M machine[:port]|:port] [-N bp] [-O a|b|c|h|m|p|s|t] [-q] [-R rootdir]
   [-s] [-S {a|t}:{on:off}] [-t]
```

Where

-a[:*aflag*]|-A *file*|*dir*[:*aflag*] (Audit)
: Enables auditing (default icnetd.lg). Where *aflag* is a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-h|-? (Help)
: Displays help text.

-d (Debug)
: Run in debug mode, no daemonization. (On Linux only.)

-L *file*|*dir*[:*aflag*] (Log Surrogates - new in 5.40)
: Enables specifying an alternate location and/or format for logging surrogates. This option overrides the option -S a:{on|off}

-M *machine*[:*port*]|[:*port*] (Machine)
: Specifies the remote machine and/or TCP service port address for ICNETD. *Machine* defaults to localhost if not specified. *Port* defaults to 7333 if not specified.

-N bp (No options)
: Specifies NO options. Valid NO options are:

b (No-logon-as-batch)	Allow logons to the icios server without the logon-as-batch privilege (On Windows only)
p (No-password)	Allow logons without passing a password (On Linux only)

-O a|b|c|h|m|p|s|t (Operation)
: Specifies an operation to perform. Valid operations are:

a (Amplify)	Amplify daemon tracing
b (Boost)	Boost (amplify) server tracing
c (Check)	Check to see ICNETD is already running
h (Hush)	Hush (mute) server tracing
m (Mute)	Mute daemon tracing
p (Post)	Cause connection information to be written to the log file
s (Start)	Start ICNETD (On Windows only)
t (Terminate)	Terminate ICNETD.

-q (Quiet)
: Enables quiet operation.

-R *rootdir* (ROOT)
: Specifies the effective root directory on the machine to which thickclient remote users have access. Default is "/" on Linux and "current-drive:\" on Windows. Only used by the icios server.

-s (Service)
: Service indicator. On Windows it is required when running as a service. On Linux it is required when running under systemd (RHEL 7 and up) vs. /etc/init.d (RHEL 6 and before, or emulated in RHEL 7 and up)

-S {a|t}:{on:off} (Server)
Server options:
a-audit (per processing)
t-tracing can be enabled and disabled.
-t (Trace)
Enables tracing to allow debugging.
-T *seconds* (new in 5.40)
Set a time limit for retaining icrunrs surrogates that are awaiting a reconnect from a client. The maximum is 65534 (a bit over 18 hours). The default is to retain them indefinitely.

Environment variables:

ICNETD	Command line options
ICPERMIT_MACHINE	Remote machine for server licensing

ICNETD is managed differently depending on whether it is configured to run under /etc/init.d or under systemd. When using /etc/init.d, ICNETD is managed as part of the ICOBOL services group using the service and chkconfig applications. When using systemd, ICNETD is managed as an independent service using the systemctl application.

You must be super user to start the ICNETD server and to use the “amt” Operate options.

ICNETD -O c (check) can be done by any client to see if an ICNETD daemon is running. The Machine switch can be used to check a remote machine.

C. **Description**

From the operating system standpoint, a TCP/IP network must be running between the client and server machines with the necessary daemons. The *ping* utility can usually be used from both the client and the server to check that each can access the other with the given host-name, although it is often blocked by default in many firewalls.

Each service request requires Network Server licenses for the various services that it offers as described in the preceding section.

ICNETD can be started and stopped using the service utility (ln6) or the systemctl utility (ln7 & ln8)..

On the server system, ICNETD must be running to provide the initial server connection to the client. When ICNETD starts, it registers itself with TCP/IP as a server listening for connections on a particular TCP/IP port. The default port is 7333, but this can be changed from the command line. When a client opens a TCP/IP connection on this port, ICNETD is notified of the connection and the type of request (either ThickClient, ThinClient, or remote ICSQL client) being made. ThickClients are from remote runtimes, ICODBC drivers, or user library applications that need client/server file support while ThinClients are specialized front-ends on the clients that provide only a very small interface handler. ICNETD then tries to log in the given username/password. Up to three attempts will be made before an error is returned to the application. If no such user is found, an exception 309 "Network path was not found" is usually given. While making these attempts, username, password, and domain will be prompted for on the client machine. Once logged in, the appropriate server (icios for ThickClient support, icrunrs for ThinClient support, or icsqls for remote ICSQL support) is started and the appropriate license(s) are requested. From this point on, the server process will handle all client requests. If ICNETD is not running when a connection is attempted, the client will usually receive an exception 315.

Each server process acts just like that user with all the same access controls and privileges. Icios servers will remain until the process that requested the service terminates. (I.E., even if all opened “remote” files are closed such that there are no open across the connection the server will remain.) This provides a performance boost for closing and then re-opening files in an application at the expense of keeping the process and license in use.

The No-password option (-N p) can be used to not require a password. For security reasons, this is not recommended.

The username/password prompt has an option to save the information for future logins. The username/password/domain will be saved for each ip address in the user's registry on Windows clients and in a file with the name .icnet.<ip-address> written to the user's home directory (as given by the HOME environment entry) for Linux clients.

The ICNETD servers icios and icrunrs require ICEXEC to be running. The ICNETD server icsqls does NOT require ICEXEC to be running.

NOTE: ICNETD servers use processes as well as file resources from ICEXEC, thus the process count and other system parameters in the system configuration must take this into account.

When ThickClient clients access files through ICNETD, the file pathnames are relative to ICNETD's effective root directory. By default, this is the actual root of the server file system. It may be desirable for security reasons to limit remote users to a subset of the server file system. This can be done by using the -R parameter when starting ICNETD to change its effective root. It will prefix the filenames from the client with the subdirectory from the -R parameter before opening them. Thus, if -R is set to "/remote/files" and the client opens "/ar/customer", the server will open "/remote/files/ar/customer".

The Rootdir switch (-R *rootdir*) instructs ICNETD to always prepend the *rootdir* to any name passed to the ICNETD server for ThickClients. (Icios)

If no Rootdir is given, all filenames start at the actual root directory.

When server tracing is enabled, each server generates its own log file (icios_<*time*>.lg, icrunrs_<*time*>.lg, ...) in the **ICOBOL** working directory in addition to the log file generated by the server (ICNETD). *Time* is the timestamp of when the process started. The location and naming convention for log files can be modified using the -L option.

Another item to note, when opening files ASSIGN'ed to PRINTER, if the filename is an ICNETD remote file it is NOT placed in the local printer control file (.pq). If the ICNETD server has an ICEXEC running with PCQ's enabled, it will be placed in that printer control queue (.pq) file. The console number for the entry will be set to -1.

On termination, ICNETD outputs a table of connection information to the log file.

D. Use as ThickClient (icios)

The ThickClient client accesses files on the server by using a special network filename. The syntax of this filename uses the special leadin character that is also used by logical device names followed by a standard Internet Uniform Resource Locator (URL). The syntax is as follows:

```
@[icnet:]//machine[:port-address]/path
```

Where

machine
: Is the remote machine name or IP address of the machine on which you wish to access files.

port-address
: Is the TCP service port on which ICNETD is listening on the remote machine instead of the default (7333).

path
: Is the filename, including any directory specifiers, to the file on the specific machine.

The *machine* is often a simple name on a local area network, e.g., "accounting". It can be a full internet name on a wide area network, such as "accounting.envyr.icobol.com", or an IP address, such as "166.82.100.101". The naming used will depend on how your network is configured.

As mentioned above, the *path* supplied will depend on whether ICNETD has been configured with an effective root or not. In order to access the file "/remote/files/ar/customer" on the "accounting" server, the client would specify the following:

ICNETD<enter> (ICNETD started with the default root)

@icnet://accounting/remote/files/ar/customer

ICNETD -R /remote/files<enter> (ICNETD started with a new effective root)

@icnet://accounting/ar/customer

In order to print a queued file on the server using @PCQ0, the client would specify:

@icnet://accounting/@PCQ0

Nothing except TCP/IP is required on the client system from Interactive COBOL to connect to the server. Client exceptions that can be received trying to connect are:

252 "Program is not authorized to run"	A Network Server license is not available on the specified machine.
306 "Network Request not supported"	A revision mismatch between the client and the ICNETD daemon.
307 "Remote Computer is not available"	There is no computer by the given machine name available on the network.
309 "Network path was not found"	The current username is not available (from /etc/passwd) on the remote machine. Access denied.
315 "Unexpected Network Error"	There is no ICNETD running on the remote machine.
323 "Network name not found"	Couldn't set group-id or user-id from /etc/passwd.

On the server system, the ICNETD daemon must be running to provide the initial server connection to the client. It then forks a server ICNETD process that "logs into the given user's account" on that machine. For this reason, all users who access files on a server must have accounts available on that server with matching user names. Each server process acts just like that user with all the same access controls and privileges. If no such user is found, an exception 309 is given. On Windows, ICNETD starts icios servers with the logon_batch option so all usernames on the ICNETD server machine must have the “Logon as a batch job” privilege when using thickclients. To add this user right do the following on the machine on which ICNET is running: Select the User manager. Select Policies and then User Rights. Check the Show Advanced User rights box. Now select the "Log on as a batch job" right and then add the needed groups and/or users.

If the username/password is invalid on the first open to the ICNETD server, the user will be prompted for a valid username/password with a pop-up box. Three(3) attempts will be allowed before an error is returned. ESC will cause the open to fail. Once a valid password is given, it will be remembered for all subsequent connections. If a new username was given, the new username/password pair is not remembered for a new connection, the original username/password pair will be used.

When communicating with a Windows based ICNETD, the following os-errors can be reported in the Logon Failure username/password pop-up dialog box. Use these error descriptions to help solve the problem.

1314	Privilege not held
1315	Invalid account name
1317	No such user
1326	Logon failure (unknown username or bad password)
1327	Account restriction
1328	Invalid logon hours
1329	Invalid workstation
1330	Password expired

1331 Account disabled
1385 Logon type not granted (Need a privilege)

From a client, to check that the ICNETD server is running on a remote machine the following should be done:

```
icnetd -O c -M machine
```

Icios can be set to detect loss of client connection by using the ICNETDUSESHEARTBEAT environment variable from the client side.

Setting ICNETDUSESHEARTBEAT=1 causes a heartbeat thread to be enabled on the client and server to continuously provide a heartbeat across the network. In this mode if icios detects a loss of the hearbeat it will shutdown the icios server process cleanly closing all files. Usually this will happen withing 60-120 seconds.

Some Examples:

On Linux, if you start ICNETD with no root directory (-R):

@//server1/usr/joe/data	would access /usr/joe/data on machine server1
@icnet://server1/usr/joe/data	would access /usr/joe/data on machine server1
@//server2/data	would access /data on machine server2
@//server2/@pcq6	would access @PCQ6 on server2

On Windows, if you start ICNETD with no root directory (-R):

@\\server1\user\joe\data	would access \user\joe\data on machine server1 drive C:
@icnet:\\server1\user\joe\data	would access \user\joe\data on machine server1 drive C:
@\\server2\D:\data	would access D:\data on machine server2
@\\server2\@PCQ6	would access @PCQ6 on server2

The ICLINK utility can be used to provide a mapping from filenames in the COBOL program to client/server type filenames. See the ICLINK Chapter for more information.

E. Use as ThinClient (icrunrs)

When icnetd starts a ThinClient server (icrunrs), it will pass the client's ip address in as an environment variable called ICREMOTEADDRESS and the client's host name as ICREMOTEHOST. (Basically a "-E ICREMOTEADDRESS=n.n.n.n -E ICREMOTEHOST=xxxx" on the command line.) These two entries can then be queried from COBOL by using the IC_GET_ENV builtin after determining that a ThinClient is running by doing an IC_TERM_STAT builtin and looking at the two ThinClient flags.

The sample logon program has been updated to show this information in the upper left corner of the main screen, if available.

The ThinClient server (icrunrs) is started by ICNETD and runs the logon program by default. On Windows, the ThinClient server is installed when ICNETD is selected. When the ThinClient server is invoked by ICNETD, it requests both a runtime license and a Network Server license from the license manager and then starts the COBOL program. If sp2 or qpr calls are made by the COBOL program then a sp2runtime license will be acquired at that point. The ThinClient server uses consoles with device set to "machine-name" or ip-address first, then "icrunrs", and

finally to (blank). The ICTERM setting is provided by the ThinClient client. Note that all users that attach to ICNETD via a thinclient must have the "Log on locally" privilege when the server is a Windows machine. Also note that the password cannot be empty.

On the server ensure that the following are accessible in the current directory or via PATH, ICCCODEPATH, ICDATAPATH, etc:

- cobol object code (.cx files)
- data files

Once the application is running, it will make user interface calls which are intercepted by the ThinClient server library. Some of these calls are processed on the server and some are sent to the client machine for processing. Normally character calls sent to the client will result in a response from the end user. Each ThinClient server (icrunrs) requires a runtime license, Network Server license, and possibly an SP2Runtime license.

To debug ThinClient consider the following:

Make sure the program(s) run without ThinClient before moving to ThinClient.
On the server

1. Configure ICNETD surrogate tracing (icnetd -O b). This will cause icrunrs_(pid).lg files to be created for each icrunrs started. Any **ICOBOL** errors will be logged to this log file. Note that the location and format of these log files can be set with the -L option.
2. Configure ICNETD server tracing (icnetd -O a). Provides more logging information in the icnetd.lg file.
3. Alternatively, e.g., when the server is in production and you don't want all the logging turned on all of the time, the enhanced logging for the server and for surrogates can be turned on and off dynamically with the -S options. Note that the -L option must be set before the service is started.

If gui support (sp2 and/or qpr) is also to be used then an additional SP2RUNTIME license is required on the server. To run totally in gui mode the logon program sp2logon can be invoked. Gui support is only provided when the ThinClient client is running on Windows.

On the server ensure that the following are accessible in the current directory or via PATH, ICCODEPATH, ICDATAPATH, ICCONFIGDIR, SP2DIR, SP2.CFG, etc:

- cobol object code (.cx files)
- data files
- panel files
- sp2 configuration file
- sp2tc.ini

Once the application is running, it will make SP2 (and FormPrint) user interface calls which are intercepted by the ThinClient (gui) server library. Some of these calls are processed on the server and some are sent to the client machine for processing. Normally sp2 calls sent to the client will result in a response from the end user. Each ThinClient server that uses SP2 or QPR requires an SP2 runtime license in addition to the standard runtime license and Network Server license when using gui calls.

To debug ThinClient with gui consider the following in addition to the steps outlined above:

1. On the client, set SP2DBG=2 to get an sp2dbg.xxx log file. (QPRLOG=1 for FormPrint)
2. On the server, set SP2DBG=2 to get an sp2dbg.xxx log file. (QPRLOG=1 for FormPrint.)

More on using ThinClient with gui support can be found in the readsp2.txt file.

F. Use as ISQL Client (icsqls)

The ISQL client accesses databases on the server by using a special network filename in the DSN specification in the CONNECT statement.. The syntax of this DSN uses the special leadin character that is also used by logical device names followed by a standard Internet Uniform Resource Locator (URL). The syntax is as follows:

```
@[icnet:]//machine[:port-address]/connection-string
```

Where

machine
: Is the remote machine name or IP address of the machine on which you wish to the database.

port-address
: Is the TCP service port on which ICNETD is listening on the remote machine instead of the default (7333).

connection-string
: Is the same connection string that would have been supplied if the database were located locally instead of remotely.

The *machine* is often a simple name on a local area network, e.g., "accounting". It can be a full internet name on a wide area network, such as "accounting.envyr.icobol.com", or an IP address, such as "166.82.100.101". The naming used will depend on how your network is configured.

Normal ICNETD login conventions apply to making the connection to *machine*. Also, the ICNETUSESHEARTBEAT environment variable is support for icsqls.

The connection and database processing are handled via ODBC calls, so an ODBC framework must be available on the server machine.

Using ISQL in client/server mode can greatly reduce network traffic and increase performance in comparison to the client using a local ODBC driver that does its processing via a network share.

G. ICNETD Auditing

ICNETD provides various logging modes to facilitate debugging.

Initially, a default ICNETD log file will look like:

```
Audit log for icnetd 4.20 (Windows) created Dec-21-2009 09:34:13.00
icnetd Revision 4.20 (Windows)
Copyright (C) 1987-2009, Envyr Corporation.  All rights reserved.
Started without Startup Parameters specified
Options: -A C:\WINDOWS\:b -s -M :7333
Dec-21-2009 09:34:13.80 icnetd (868):          icnetd is ready, listening on: port=7333 on machine RALPHJ
```

When running with ThinClients that happen to disconnect for some reason the following will be added:

```
Dec-21-2009 09:51:49.26 icnetd (868):          Reconnect request from surrogate on pid 2064 for client on
machine ENVYRMOBILE pid 2320
Dec-21-2009 09:51:58.81 icnetd (868):          Reconnect request from surrogate on pid 1676 for client on
machine ENVYRMOBILE pid 2884
```

These could reconnect or not as:

```
Dec-21-2009 09:53:49.60 icnetd (868): Warning: Reconnecting surrogate terminated (Exit Code = 0): pid=2064 for
client on machine ENVYRMOBILE pid 2320
Dec-21-2009 10:58:41.11 icnetd (868):          Reconnecting client on machine ENVYRMOBILE pid 2884 to surrogate
on pid 1676
Dec-21-2009 10:58:41.29 icnetd (868): Error:   Reconnecting from 75.251.64.189:57346
Dec-21-2009 10:58:41.29 icnetd (868): Error:   The reconnection key does not match any connection: Processing
reconnect
```

In one case (surrogate 2064) you will see that the surrogate terminated, in this case ICREMOTETIMEOUT had been set and it timed out.

In the second case (surrogate 1676) no timeout was given and the remote client reconnected.

The two error lines are when the thinclient client tried to reconnect with the previously terminated surrogate (2064).

Additional lines in the log can be shown by making an ICNETD -O p (post) call. This will cause ICNETD to dump its current connection table as such:

```
Dec-21-2009 11:05:34.11 icnetd (868):          Connected to server (command request) 127.0.0.1:2933
Dec-21-2009 11:05:34.11 icnetd (868):          Post current connection data:
 Client........ .............. .... ........ ............. ....... .... Surrogate  Date and Time
 Computer Name  IP Address     Port      PID User Name     Program Rev       PID Connection Established  Recon
 -------------- -------------- ----- ------- ------------- ------- ---- -------- ----------------------- -----
  ENVYRMOBILE   75.251.64.189  57347    2828 Ralph Jordan  icrunrc  4.20     3652 Dec-21-2009 11:03:09.00    No
  ENVYRMOBILE   75.251.64.189  57345    2884 Ralph Jordan  icrunrc  4.20     1676 Dec-21-2009 09:50:23.00    No
  RALPHJ        192.168.2.105   2930     696 ralph         icrun    4.20     1800 Dec-21-2009 11:05:23.00    No
 -------------- -------------- ----- ------- ------------- ------- ----- -------- ----------------------- -----
```

Note the final column of Recon. This column will be set to “Yes S” if the surrogate has requested a reconnection or an “Yes C” if the client has requested to reconnect.

By adding additional auditing options this log can be used to show all the actual requests and even the command lines used to start the various surrogates. ICNETD -O a, ICNETD -O b, etc. All of these operations/options can be set with either the command line directly when starting ICNETD or as operations from a command ICNETD. These should only be used for trace and debugging purposes as they will create a much larger log file.

VIII. THINCLIENT

A. Introduction

ThinClient support is provided to allow an application to be run on a server (either Linux or Windows) and deploy the application's user interface to any client machine connected to the server via a TCP/IP network. The clients can be any Linux or Windows based machines for which an icrunrc executable is provided. When running in Windows the icrunrc executable provides both character-based and gui-based application support.

The main advantage to the thin-client architecture (versus the standard thick-client architecture which is done via a runtime accessing data files over the network) is that the business logic and database files are maintained and executed in a central location yet end-users still have access to the user interface and all its associated conveniences. Only the standard **ICOBOL** screen interfaces (or gui sp2/FormPrint interfaces) will be shipped to the client. The COBOL program itself runs on the server machine and accesses files directly on that machine. If the client is running under Linux, NO gui Sp2 and/or FormPrint interfaces will be available.

Think of the ICRUNRC/ICRUNRS combination as a type of telnet session with ICRUNRC being the telnet client/terminal emulator and ICRUNRS being the telnetd server connection.

The TCP/IP connection between the ThinClient client (icrunrc) and the ThinClient server (icrunrs) is encrypted and has reconnection handling.

On Windows, no telnetd server must be purchased for a Windows server machine to support client connections.

On Windows, no terminal emulator client must be purchased to connect to Linux or Windows server machines when using the ThinClient client. The ThinClient client works just like the thick-client icrunw runtime.

B. Environment Entries

B.1. Overview

ICRUNRC searches for specific Interactive COBOL entries in the user's current environment. These entries allow the user to tailor a particular session of a particular terminal type.

Interactive COBOL environment entries other than ICROOT and ICCONFIGDIR that are used are:

ICBGCOLOR	Specify the initial background color
ICCOLOR	Specify how to support color
ICCOLUMNS	Columns for terminal
ICFGCOLOR	Specify the initial foreground color
ICLINES	Lines for terminal
ICREVERSE	Specify how to support reverse
ICRUNRC	Default switches for icrunrc
ICSCROPT	SCREEN OPTIMIZER selection
ICTERM	Terminal type
PTS	Print Thru device
ICRECONNECTTIMEOUT	Timeout in seconds before reconnecting is terminated

When using the Bourne shell these entries must be exported to be made available to ICRUNRC.

B.2. ICBGCOLOR, ICCOLOR, ICFGCOLOR

ICCOLOR specifies how to interpret color codes from a COBOL program. Valid selections are filter, ignore, and process; the default is filter.

The syntax is:

```
ICCOLOR=filter|ignore|process
```

Where

filter
: Causes the client to watch for color codes from the program and to NOT send them to the terminal, since it does not support color. Filter is the default.

ignore
: Tells the client that the user wants total control of the screen and may be sending binary color data to the screen and that the client should ignore all color codes (i.e., do not look for color codes). If running in this mode, the SCREEN OPTIMIZER should not be enabled as the client cannot accurately repaint a user's screen.

process
: The client interprets color codes from the program and sends the appropriate sequences to the terminal. When set to Process the initial background and foreground colors are set by the client at startup.

ICBGCOLOR sets the default background color to the indicated value when running with ICCOLOR set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7) either as the name or the number. The default is black (0).

The syntax is:

```
ICBGCOLOR=black|blue|green|cyan|red|magenta|brown|white|
            0|1|2|3|4|5|6|7
```

ICFGCOLOR sets the default foreground color to the indicated value when running with ICCOLOR set to Process. Valid selections are black (0), blue (1), green (2), cyan (3), red (4), magenta (5), brown (6), and white (7) either as the name or the number. The default is white (7).

The syntax is:

```
ICFGCOLOR=black|blue|green|cyan|red|magenta|brown|white|
            0|1|2|3|4|5|6|7
```

Currently only pcwindows, DG, and Ansi based terminals support color.

B.3. ICTERM, ICCOLUMNS, ICLINES

ICTERM specifies what type of terminal is being used. ICLINES and ICCOLUMNS specify the number of lines and columns for the terminal.

The syntax is:

```
ICTERM=terminal-type
ICLINES=lines
ICCOLUMNS=columns[:ccolumns]
```

Where

terminal-type
: Specifies a valid ICTERM entries described below or in the ICTERM Chapter starting on page 125, with a corresponding terminal description file

lines
Specifies the number of lines for this terminal. It can range from 24 to 255.

columns[:*ccolumns*]
Specifies the number of columns for this terminal in its default mode. It can range from 80 to 255. If the second value is given, provides a secondary number of columns that are supported. The lowest value is considered the normal mode while the highest value is considered the compressed mode. Basically it can be min:max or max:min.

If not specified in the environment then terminfo is used.

If ICLINES and/or ICCOLUMNS are not specified, or are 0, then the numbers in the appropriate terminal description file are used except for the terminfo and pcwindow terminal descriptions. For terminfo descriptions, ICLINES and/or ICCOLUMNS are those defined by the LINES and/or COLUMNS from the terminfo database. For pcwindow descriptions, ICLINES and/or ICCOLUMNS are those defined by the video bios. If no ICLINES or ICCOLUMNS information can be found, the defaults of 24 lines by 80 columns are used.

Current valid default ICTERM terminal-types (all lower-case) and a brief description are given below:

aix	the high function console (hft) on an AIX machine,
ansi	a standard ansi terminal (like a vt100 but no function keys),
att	an AT&T 605 type terminal,
dg	a DG D200 or above type terminal,
dgunix	a DG D217+ terminal in dgunix mode
freedom	a Freedom ONE type terminal,
terminfo	any terminal defined in the terminfo database,
ibm	an IBM 3101 or 3151 type terminal,
linux	a Linux master console,
mac	a Terminal session on Apple Mac OS X,
sun	a SUN master console,
vt100	a VT100 type terminal,
vt220	a VT220 type terminal,
vt220pc	a VT220 emulation using a pc keyboard,
wyse	a Wyse 50-plus type terminal,
wy50	a Wyse 50 type terminal with NO attribute support,
sco, xenix	an SCO master console or SCO master console type terminal,
xterm-fk	an XWINDOW terminal emulator
386ix, at386	a 386/ix or AT&T 386 master console

More on these terminal types can be found in the ICTERM Chapter, starting on page 125.

ICLINES should be set to the line at which the terminal will scroll the screen when a line-feed (<lf>) is sent. (Line-feed and newline are the same.)

ICCOLUMNS should be set to the column position after which the terminal will wrap to the next line. If both normal and compressed spacing is available then two values should be specified separated by a colon (:) with the first being the default mode.

If a terminal is supported directly by Interactive COBOL, it should be used instead of using terminfo because Interactive COBOL generally provides better performance and memory usage.

B.4. ICREVERSE

ICREVERSE specifies how to interpret reverse codes from a COBOL program.

The syntax is:

```
ICREVERSE=filter|ignore|process
```

Where

filter
: Causes the client to watch for reverse codes from the program and to NOT send them to the terminal, since it does not support reverse.

ignore
: Tells the client that the user wants total control of the screen and may be sending binary reverse data to the screen and that the client should ignore all reverse codes (i.e., do not look for reverse codes). If running in this mode, the SCREEN OPTIMIZER cannot correctly repaint a user's screen that includes reverse.

process
: The client interprets reverse codes from the program and sends the appropriate sequences to the terminal. Process is the default.

B.5. ICRUNRC

The contents of the *ICRUNRC* environment variable are treated like switches entered from the command line and processed before any other switches or arguments when starting ICRUNRC.

The syntax is:

```
ICRUNRC=icrunrc-switches
```

Where

icrunrc-switches
: Specifies any valid command line switches for ICRUNRC but not a command-line argument.

B.6. ICSCROPT

ICSCROPT specifies whether to enable the Interactive COBOL SCREEN OPTIMIZER.

The syntax is:

```
ICSCROPT=off|on|full|partial|mute
```

Where

off
: Disables screen optimization

partial
: Enables simple single screen optimization

on and full
: Enables full screen optimization

mute
: Disables any screen optimization and prevents any implied codes to be sent at startup or termination.

The SCREEN OPTIMIZER keeps track of all data sent to the console and prevents rewriting the same data multiple times. It uses an image of the current screen always in memory.

Usually screen optimization will provide improved screen performance. **The Screen optimizer should ALWAYS be enabled on the server side.**

The full option usually provides better screen update performance than the partial option, but it requires more memory and cpu time.

B.7. PTS

PTS allows a specific device or output-pipe to be set for printer-pass thru in this session. If set, this interception is done BEFORE the data is sent to the particular terminal.

The syntax is:

PTS=*device* | *output-pipe*

Where
output-pipe
is an output pipe with a leading bar greater than sign as: "|>lp"

For example:

```
PTS=/dev/lp
export PTS
```

or

```
PTS="|>lp"
export PTS
```

The first would cause the device /dev/lp to be opened and all data sent to that device.

The second will pipe the output to the lp command using the default printer.

Note that the bar and greater than sign must be escaped to prevent the shell from processing them.

B.8. ICRECONNECTTIMEOUT

ICRECONNECTTIMEOUT allows a specific time in seconds to be specified for ICRUNRC to continue to try to reconnect to the server when a connection has been lost.

The syntax is:

ICRECONNECTTIMEOUT=*seconds*

Where
seconds
is from 0 - 65535.
0 is wait for ever, the default
1-6300 is to wait for that long in seconds (6300 is 1 hr and 45 minutes)
>6300 is set to 6300.

If the connection between the ThinClient client and server is dropped for some reason the ThinClient client will display a Reconnecting message similar to below:

```
                Reconnecting
------------------------------------
Connection reset by peer (oserr=10054)
Press ESC to cancel and exit icrunrc
```

or

```
                Reconnecting
----------------------------------------
The network heartbeat has failed to respond
Press ESC to cancel and exit icrunrc
```

The ThinClient client will continue to try to reconnect. When it is successful the message will be removed from the screen and the session can be continued.

C. Syntax

The syntax is:

```
icrunrc [-a[:aflag] | -A file|dir[:aflag]] [-B 32|64] [-c] [-E var=val][-h|-?] [-i]
    [-m|-M machine[:port]] [-N swx ] [-p] [-q] [-t]  -- icrunrs-switches ] [ program
    [arg]... ]
```

Where icrunrc switches are:

-a[:*aflag*] or -A *file*|*dir*[:*aflag*] (Audit)
: Enables auditing (default icrunrc.lg). Where *aflag* can be a|b|d|p|t|u|da|db|pa|pb|ta|tb|ua|ub, defined as a-append, b-backup, d-date, p-pid, t-time, and u-username.

-B 32|64 (bitness)
: Specifies which surrogate (32 or 64 bit) to start. The default is the bitness of the client.

-c (Crash recovery)
: Enable Crash recovery mode.

-E *var*=*val* (Environment)
: Set the environment variable *var* to *val* in the current environment.

-h | -? (Help)
: Display help text.

-i (Info)
: Display additional informational messages

-m (Prompt for machine)
: Prompts for a machine name.

-M *machine*[:*port*]] (Machine)
: Specifies the remote machine as an ip-address or machine-name and an optional port to connect to. If *port* is not given, 7334 is the default. If *machine* is not given, then localhost is used.

-N {w}... (No warnings)
: Specifies a NO option: Valid options:
  w - No warnings messages are generated.

-p (Prompt for username)
: Always Prompt for username to pass to the server.

-q (Quiet)
: Enables quiet operation.

-t (Trace)
: Enables trace information to be written to the audit log. Useful for debugging. When combined with the Info switch even more trace information is given.

-u|-U *username* (Username)
: Prompt for username (-u) or pass the given username to the server.

-- (Start Icrunrs switches)
: Two dashes (--) instructs the ThinClient client to pass all remaining switches and arguments to the ThinClient server (icrunrs).

Program

The COBOL program to start. If not specified, then the program specified in the server's configuration file is used. If that is not set, then logon is used.

Arg

Any arguments needed by *program*.

When using the Prompt for username switch, the Save option is ignored in the username/password screen. For thin clients the prompt is done immediately.

Where icrunrs switches are:

-C l|n|u (Case)
Specifies the default Case conversion (Linux):
l=lower, n=none, u=upper (default is l)

-D *date*[:*time*] (Date/time override)
Specifies an override for the default date/time. Date/time formats are YYYYMMDD & HHMMSS

-E *var*=*val* (Environment)
Sets the environment variable *var* to *val* for icrunrs

-G drsu (General)
Specifies a General switch:
d=duplicate key I-O Status, r=RDOS-like ACCEPT, s=strict switch processing,
u=uppercase program name in ACCEPT FROM ENVIRONMENT

-i (Info)
Put out Info messages

-I 2 (**ICOBOL** 2)
Run **ICOBOL** 2 .cx files. This is no longer required as the runtime will auto detect an **ICOBOL** 2 .cx.

-M *mode* (window Mode)
Specify the initial window mode for the runtime, passed to a Windows icrunrc Valid values for *mode* are the values for the IC_WINDOWS_SHOW_CONSOLE builtin plus 0. 0 does nothing, 1 is Hide, 2 is Maximize, 3 is Minimize, etc. Only useful when communicating to a Windows client.

-N beinow (No)
Specifies NO options:
b=No auto reassign, e=No embedded spaces, i=No console interrupts,
n=No Nagle Algorithm suppressed, o=No OCCURS DEPENDING bounds check,
w=No warnings

-p (Prompt for username)
Always Prompt for username to pass to the server via ICNETD.

-q (Quiet)
Specifies Quiet operation

-s (Startup mode)
Run *program* in startup mode

-S *n* (System code)
Specifies the System code value to be returned in ACCEPT FROM ENVIRONMENT

-T *n* (Terminal)
Specifies the Terminal number is *n*

-U l|n|u (Username)
Convert username case:
l=lower, n=none, u=upper (default is n)

-z | -Z *dir* (Debugger)
Run debugger, use current or specified *dir* for .sy files

ICRUNRC Environment variables:

ICCONFIGDIR	Configuration information		
ICRUNRC	Command line options	ICREVERSE	Reverse-video Control
ICTERM	Terminal description	ICSCROPT	Screen Optimizer mode
ICLINES	Screen height	ICCOLUMNS	Screen width

ICCOLOR	Color handling mode	ICBGCOLOR, ICFGCOLOR	Color values
ICFONT (win)	GUI font	ICFONTSIZE (win)	GUI font pointsize
PTS	Print pass thru device	ICRECONNECTTIMEOUT	timeout seconds

ICRUNRS Environment variables:

ICRUNRS	Command line options	ICPCQDIR	Print job file directory
ICRUNLK	File name link file	PRN,SER,PCQ	Default device units
ICCODEPATH	.CX file path list	ICDATAPATH	Data file path list
ICTIMEOUT	Timeout Delay	ICABORT	Abort on Timeout
ICPCQFILTER	PCQ filter command	ICPERMIT_MACHINE	Servermachine[:port]
DATAFILE	@DATA resolution	LISTFILE	@LIST resolution
ICSDMODE	Screen Handler Mode	ICTMPDIR	Temporary file directory
ICRECONNECTTIMEOUT	timeout seconds	ICPROMTCHAR	Screen prompt character

The default ICRUNRS environment variables are from the ICNETD - ICRUNRS process tree.

When ICNETD starts a ThinClient server (icrunrs), it will pass the client's ip-address and host-name in the environment variables ICREMOTEADDRESS and ICREMOTEHOST respectively. (Basically `"-E ICREMOTEADDRESS=n.n.n.n -E ICREMOTEHOST=machine"` on the client command line.) These can then be queried from a COBOL program by using the IC_GET_ENV builtin after determining that a ThinClient is running by doing an IC_TERM_STAT builtin and looking at the two thinclient flags (SP2-SRV-FLAG and CHAR-SRV-FLAG).

The sample logon program does look for these variables when it detects a ThinClient.

More information on IC_TERM_STAT can be found in the **ICOBOL** reference manual. There are additional builtins IC_CLIENT_CALL_PROCESS, IC_CLIENT_DELETE_FILE, IC_CLIENT_GET_ENV, IC_CLIENT_GET_FILE, IC_CLIENT_PUT_FILE, IC_CLIENT_RESOLVE_FILE, IC_CLIENTSET_ENV, IC_CLIENT_SHELLEXECUTE that can be used to communicate with the client machine. More information on these can be found in the **ICOBOL** Reference Manual.

Note that on Linux machines, the *machine-name* is sought in a case-sensitive fashion and Windows machines pass an uppercase *machine-name* while Linux machines pass a lowercase *machine-name*.

The trace switch, -t, requires the audit option.

When used with the Tracing switch (-t), the Info switch (-i) causes more trace info to be dumped to the audit log.

D. ThinClient Features

The ThinClient has an automatic reconnection ability when attached to an ICNETD/ThinClient server of 4.00 or above.

If the connection between the ThinClient client and the server is dropped for some reason the ThinClient client will display a Reconnecting message similar to below:

```
                Reconnecting
 ------------------------------------
 Connection reset by peer (oserr=10054)
 Press ESC to cancel and exit icrunrc
```

or

```
                Reconnecting
 ----------------------------------------
 The network heartbeat has failed to respond
```

```
Press ESC to cancel and exit icrunrc
```

The ThinClient will continue to try to reconnect. When it is successful the message will be removed from the screen and the session can be continued.

If at any time you wish to terminate the reconnection process just hit ESC.

The ThinClient connection provides for compression on the downstream leg. This should help performance on slower network connections.

The ThinClient server, icrunrs, has an additional No switch. The "-N n" switch will cause the socket connection from the server (icrunrs) to the client (icrunrc) to have the TCP Nagle Algorithm NOT be suppressed. In some cases, this may help screen performance over a slower WAN connection.

In the default case, the Nagle Algorithm is always disabled.

This option can be passed to icrunrs from icrunrc as:

```
icrunrc -M server-machine -- -N n
```

This option should be used with care as it has the tendency to slow down performance on an LAN. If auditing is enabled for the servers, an Info message is written to the audit log if info has also be enabled.

ICSCROPT=full should be enabled for both the icrunrc client and the icrunrs server to provide the best screen buffering options, especially over a slow WAN connection.

On the client machine, the ICRUNRC command will invoke the server version of the runtime thinclient(ICRUNRS). All screen input and output will be performed by this thin client.

On the server machine, an ICRUNRS will be started by ICNETD to run COBOL programs for that particular ThinClient session.

The ICRUNRC uses all the normal screen environment setting to set up its screen capabilities for the client. These include ICTERM, ICSCROPT, ICLINES, ICCOLUMNS, etc. .

The ICRUNRS server requires a license, called the Network Server (ICNET) in addition to an standard Runtime license. The ICRUNRS server will run under control of ICNETD under both Linux and Windows. If gui (sp2/qpr) support is used then an SP2RUNTIME license is also required.

When using ThinClient, the environment for ICNETD is very important as it is what is passed to the ICRUNRS startup. Especially important is ICCODEPATH, etc.

When ICRUNRS starts, it requires a console that has been enabled and has the device name of a matching *ip-address*, a matching *machine-name*, <blank>, or icrunrs set to run programs.

To use ICRUNRC you must communicate with an ICNETD from 3.30 or higher.

The ThinClient character (icrunrc) DOES NOT use ICEXEC or require a license on the client itself.

On Linux, a very small install can be done such that only the executable icrunrc, the messages directory with system.ms, and the term directory with any needed terminal description files are all that are required. Icconfig can be used to change/update/add terminal description files.

When using ICRUNRC on Windows, the IC_WINDOWS_SETFONT builtin can be used even when connected to a Linux server to change the client font and font size.

On Windows, the user to be logged on must have the "Log on locally" privilege.

On Linux

The ThinClient client provides seamless pass - thru support for whatever terminal type is used under the thinclient client as given by the ICTERM entry.

On Windows

The ThinClient client supports the Windows GUI environment via the ICTERM=pcwindow screen interface much as the standard GUI runtime.

The ThinClient supports Print-Pass Thru by using the default Windows printer at the time of the Print-Pass Thru On. If no default printer or an error is received, then the data will go to the screen.

The Print Screen command is supported in the same fashion.

E. Using ThinClient

ThinClient client

In the Windows install, a ThinClient client option is presented to allow the ThinClient client portion of the runtime to be installed for those clients that need ThinClient access to a server computer. You are prompted for a server machine. Only Local installs are allowed.

When the ThinClient client (icrunrc) starts, it connects to the ThinClient server process (icrunrs) on a server machine running ICNETD. (icrunrc takes a -M machine argument to specify the server machine.). The client then performs a logon to the server with a username/password just as if you were accessing a file using the ICNETD file client/server support. After a good logon, the ICNETD server will start a ThinClient server (icrunrs) to provide screen-character communication with the ThinClient client.

On the client ensure that the following are accessible in the current directory or via PATH, ICROOT, etc:

- terminal description file (.tdi)

No licensing is required on the client side.

ThinClient server

The ThinClient server (icrunrs) is started by the ICNETD daemon and runs the logon program by default. On Windows, the ThinClient server is installed when ICNETD is selected. On Linux, the ThinClient server is installed by default. When the ThinClient server is invoked by ICNETD, it requests a Network Server license and an **ICOBOL** runtime license from the license manager and then starts the COBOL program logon.cx by default. The ThinClient server uses consoles with device set to a matching *ip-address*, a matching *machine-name*, <blank>, or icrunrs and set to run programs. The ICTERM setting is passed from the client. Note that all users that attach to ICNETD via thinclient must have the "Log on Locally" privilege when the server is an Windows machine. Also note that on Windows the password cannot be empty.

On the server ensure that the following are accessible in the current directory or via PATH, ICCODEPATH, ICDATAPATH, etc:

- cobol object code (.cx files)
- data files

Once the application is running, it will make console user interface calls which are intercepted by the ThinClient server library. Some of these calls are processed on the server and some are sent to the client machine for

processing. Normally console calls sent to the client will result in a response from the end user. Each ThinClient server (icrunrs) requires a Network Services license, an **ICOBOL** runtime license, and possibility an SP2Runtime license.

To debug ThinClient, consider the following:

A. Make sure the program(s) run without ThinClient before moving to ThinClient.
B. With ThinClient
   B.1 On the server, turn on ICNETD server tracing (icnetd -O b). This will cause icrunrs_(pid).lg files to be created for each icrunrs process started. Any **ICOBOL** errors will be logged to this log file. Without this log file, all **ICOBOL** messages are lost.
   B.2 On the server, turn on ICNETD server tracing (icnetd -O a). Provides more logging information in the icnetd.lg file.

To use the – switch to pass information to the ThinClient server, consider you are going to a Linux machine called aix2 and you want to support mixed cases filenames then you can do the following:

```
icrunrc -M ln6x32vm -- -C u
```

In this case, the -M ln6x32vm is processed by the ThinClient client (icrunrc) while the -C u is passed to the server and to the ThinClient surrogate (icrunrs) to be processed.

IX. OPERATING TIPS

A. Overview

Interactive COBOL has been designed to be configurable such that small systems are not burdened with large system requirements and large system are not limited by the small systems constraints. Generally the default configuration options provided by Interactive COBOL allow a basic system to be started. The tips below should be taken as a starting point to configuring a well performing system that does not include wasted memory but does allow some flexibility for growth.

B. Interactive COBOL

To get the very best possible performance, try the following tips:

1) Use the fewest possible selections in ICCODEPATH with a library as the first or second selection. (helps CALL & CALL PROGRAM)

2) Provide from 50 to 200KB per process of Interactive COBOL buffers (process local buffer cache). (helps all I/O)

3) Use a COBOL library file for all your programs, at least the most often used ones. (helps CALL & CALL PROGRAM)

4) Run terminals at the highest possible baud rates that the terminals and/or cables and/or cable length can handle. (general performance)

The System Information utility under the runtime should be viewed to help select parameters in the System Parameter menu of ICCONFIG that provide only the needed resources and no more to conserve memory.

Various Linux tools are available to monitor the status of how the operating system is performing. These include *sar* and *ps* to name a few. Particular attention should be paid to the number of buffers. Under Linux use the *top* utility to view on-going statistics.

C. Linux Configuration Tips

There are many different releases of Linux available today. Each flavor has its own conventions and so you should always consult your operating system and machine manuals before trying new things and whenever something unexpected seems to happen.

Our standard build and test environment is CentOS, which is the open-source flavor based on RedHat Enterprise Linux.

In most cases, the default kernel configuration parameters are sufficient to run all but the largest Interactive COBOL installations. If you are running a large user-count system, then the following are some Linux Kernel configuration parameters that may need to be changed to allow a particular Interactive COBOL system to run properly. The command *sysctl* can be used to set certain operating system parameters like *shmmax*. See the documentation for *sysctl* for more information on setting kernel parameters.

The readic.txt file may have more current information on these and other Linux kernel configuration parameters.

Not all parameters are available or configurable on various versions of Linux.

Important configuration parameters:

NOFILES is the maximum number of files each process can have open at any one time. Some Linux implementations default this number to twenty (20). For most Interactive COBOL systems this number should be much larger. Remember, each ICISAM file open in a program uses two system files. When Interactive COBOL initially starts, three files are used for standard input, standard output, and standard error. Another is used for the COBOL library file if specified, and one is needed to read .CX files when not found in the COBOL library file. Thus Interactive COBOL will initially claim five before any file is opened by a COBOL program. The minimum for this value is twenty (20) and the maximum is generally one hundred (100) or more. Sixty (60) is usually a good starting point for this parameter.

NFILE is the maximum number of open files at any one time for the entire system. For Interactive COBOL this should be in the range:

(NOFILES * the maximum number of concurrent Interactive COBOL users).

Generally when NFILE is changed NINODE must also be changed as it is the maximum number of active inodes.

NCALL specifies the maximum number of call-out table entries. Generally this value should be set to at least the number of active connections on the system.

SHMMAX is the maximum size of a shared memory segment allowed in the system. ICEXEC will not execute if this value is set too low. It will give a message indicating more shared memory is required and say how much total is needed. This number should then be set to that value or greater. A good beginning value is to have at least 4 megabytes.

SHMSEG is the maximum number of attached shared memory segments per process. This must be at least 1. To check that the SHMMAX and SHMSEG values are correct, do the following:

Run ICCONFIG and build a default configuration file by executing the Save command.

Now execute ICEXEC as super user. If you get shared memory errors, use the error message as a guide to changing one of the shared memory parameters to the necessary value otherwise the shared memory parameters are enough to run this particular Interactive COBOL configuration.

SEMMNU is the number of undo structures in the system for semaphores. This number must be set to the maximum number of Interactive COBOL processes you can have or greater. Running out of this parameter with Interactive COBOL will cause the message:

"Out of (disk) space: Open/Close Semaphore" or
"Out of (disk) space: Logon/Logoff Semaphore"

to be generated when starting an Interactive COBOL executable. The "Out of (disk) space" part of the message is the Linux error when trying to attach this process to the particular semaphore indicated and there are no more resources (SEMMNU structures) to perform this operation.

FLCKREC specifies the maximum number of locks that can be handled by the system. This should be set to at least twice the maximum number of Interactive COBOL users on the system and three times would not be unreasonable.

NPROC specifies the maximum number of processes allowed for the system.

MAXUP specifies the maximum number of concurrent processes a non-super user is allowed to run. In many cases a single process needs two or more additional processes to make a system call. This is especially true at startup. Running out of this value can show up as a File Status 30 in some cases. If the system is setup to allow all (or many) users to log on with the same name, i.e., op, then this value should be increased, maybe dramatically, to keep from running out of processes.

SEMMNI, SEMMNS, and SEMUME are all semaphore parameters that must be set to greater than 6.

Less important configuration parameters:

NAUTOUP specifies the buffer age in seconds for automatic file system updates. This parameter and BDFLUSHR work together.

BDFLUSHR specifies the rate in seconds for checking the need to write the file system buffers to disk. Lower values allow for more consistency on power-fails and crashes at the expense of slower performance due to the additional I/O overhead.

MAXUMEM is the maximum sizes for a user's executable programs. It should be checked to make sure it is large enough for the runtime system and utilities.

ULIMIT is the maximum sizes for a user's files. It should be checked to make sure it is large enough for the data files to be accessed.

D. Character Set

On screen input, by default Interactive COBOL allows any non-control code as a valid character, this includes the 8-bit version of the control codes. Thus characters <040> - <177> and <240> - <377> (octal) are valid characters.

On screen output, Interactive COBOL assumes DG compatible control codes and thus converts the following functions to the appropriate sequences for the specified terminal. The codes interpreted are Reverse ON/OFF (both forms), Bell, Cursor Home, Erase EOL, Erase Screen, Erase to end of Screen, Blink ON/OFF, Scroll ON/OFF, Cursor-Up, -Left, -Right, -Down, Print Pass THROUGH ON/OFF, Underline ON/OFF, and Dim ON/OFF. These are defined on page 125 under ICTERM=dg. Otherwise all characters that Linux allows are supported.

The tab character is sent to a screen as a space unless the line is in binary mode. All other unknown and unprintable characters are sent to the terminal unchanged although the cursor is not moved.

E. Backup

Backup is a very necessary function to insure that data is not lost and/or to allow for a quicker recovery from machine and disk problems that may have corrupted your data.

One item to note is that for your most frequent backups you do not need to backup those files that do not change. Obviously the most important files are those that change on a daily basis because of additions or deletions.

All files should be backed up periodically.

Backups and Restores should not be run while COBOL files are in use, otherwise files could be in an inconsistent state.

F. Symbolic links

When opening files, Interactive COBOL will always open the resolution file of the symbolic link.

For DELETE FILE, Interactive COBOL will delete the symbolic link itself, not the resolution file.

G. Delete Protection

On Linux, files cannot be individually delete protected. To delete protect a file on Linux, you must NOT have write (W) permission to the directory in which the file resides. If a directory has no write access, you can not create or delete files in that directory.

H. Use of the lp command

Interactive COBOL uses the following sequences to communicate to the Linux printer spooler.

The following is used to print a file, and an exit code of zero (0) says that the file was accepted for printing:

```
lp [-d<pname>] [-n<copies>] [-t<title>] [-o nobanner] [-q<priority>] [-w] [-c]
   [<printfile>]
```

Where

pname
Is the printername (sometimes referred to as destination)

copies
Is the number of copies

title
Is the simple part of the filename as the title

priority
Is the requested priority

printfile
Is the file to be printed

-d	used if a nonblank *printername* is used (i.e. not the default printer)
-n	used if copies > 1
-o *nobanner*	used if no banners were specified
-q	used if a non-default priority was specified
-w	used if the notify option was specified (not supported in all cases)
-c	used if the delete after printing option was specified
printfile	used with the Printer Control Utility to print an entire file, otherwise a pipe is initiated to *lp*

The following is used to check the status of the default printer assuming an exit code of zero if the queue is available:

```
lpstat -d
```

The following is used to check the status of a particular printer queue assuming an exit code of zero (0) if the queue is available:

```
lpstat -p<pname>
```

The following are used for the indicated purposes:

```
lp -i req-id -Hhold     Hold jobs
lp -i req-id -Hresume       Resume held jobs (Not on Linux)
lp -i req-id -Hrestart  Resume held jobs (Used on Linux)
```

`cancel` *`req-id`*	Cancel jobs
`lpstat -o` *`destination`*	Check the status of queued jobs for each destination (req-id's are scanned for in the *lpstat* output)

I. Setting defaults for non-terminal serial devices

One of Linux's problems is setting the attributes on a serial line that is not being used as a terminal (i.e., a terminal is a line on which a getty (or ttymon) is initially posted for the login name and password). Linux device drivers tend to set all of these type lines to one default setting that is usually not what you expect.

The current attributes of a particular device can be viewed by doing the following stty command on the appropriate device (ttyxx):

```
stty -a </dev/ttyxx
```

To remedy this situation, place a line similar to the following line in the startup script (usually /etc/rc).

```
(stty baud ixon ixany -ixoff; while : ; do sleep 3600 ; done) </dev/ttyxx&
```

Where
baud is the appropriate baud rate, i.e., 1200, 2400, 9600, etc.

This opens the necessary asynchronous line selecting the appropriate default settings (baud ixon ixany -ixoff, etc.) for the line that you wish and keeps the line open so these settings will remain in effect until the system is shutdown.

Another note, if you wish to open a tty line for binary i/o that is not your own terminal, you should use stty -a to check that the appropriate attributes are set on that line. If not, you must do something like the above stty line to set the appropriate attributes on that line. Especially be aware of the 'min' and 'time' attributes.

X. ICTERM DESCRIPTIONS

A. Overview

Following is a definition of how the default terminal description entries are setup. These default definitions are stored in ICCONFIG and the runtimes. Interactive COBOL uses the ICTERM entry to instruct it as to how to handle input and output to terminal lines. ICCONFIG can be used to change and add to these entries. If an entry is modified in any fashion, it is recommended that a new entry be built with a suitable name and comment noting that it differs from the defaults provided.

Included in each section is a basic input (from the keyboard) and output (to the display) section. More detail can be found by using ICCONFIG to view each terminal description or by printing a copy of the terminal description.

OUTPUT

On output the following DG codes are intercepted and cause the appropriate action on the terminal chosen.

Code(s)	Action	Code(s)	Action
\002	Reverse off	\030	Cursor right
\007	Bell	\031	Cursor left
\010	Cursor home	\032	Cursor down
\011	(tab) Space	\034	Dim ON
\012	Newline	\035	Dim off
\013	Erase to eol	\036A<n>	Set FG color
\014	Erase screen and home cursor	\036B<n>	Set BG color
\015	Carriage return	\036D	Reverse ON
\016	Blink ON	\036E	Reverse off
\017	Blink off	\036F'	Print Pass Through ON
\020	Position cursor col,line	\036F?2	Print Pass Through off
\021	Print Screen	\036F?3	Print Pass Through ON
\022	Scroll ON	\036Fa	Print Pass Through off
\023	Scroll off	\036FE	Erase screen and home cursor
\024	Underscore ON	\036FF	Erase to end of screen
\025	Underscore off	\036FJ	Select normal spacing
\026	Reverse ON	\036FK	Select compressed spacing
\027	Cursor up		

Where

color *n* is Ascii:

0 - black, 1 - blue, 2 - green, 3 - cyan, 4 - red, 5 - magenta, 6 - yellow, 7 - white.

Notes:

1. Print Pass Through off when going to a non-DG terminal will not generate a Ctrl-F when the printing is finished. This is strictly a DG terminal function.

2. A newline in DG causes the cursor to move to the first column of the next line.

3. Color codes are only interpreted if ICCOLOR has been set to process. (Not the default.)

4. Reverse codes are always interpreted unless ICREVERSE has been set to Ignore or Filter.

5. If a particular terminal cannot handle an attribute (either a color or character attribute, or both) then that attribute will appear to be ignored.

6. DG terminals default to Bright mode with an attribute for DIM. Most other terminals default to Dim mode with an attribute for BRIGHT.

INPUT

The following input keyboard codes are common to all terminals unless specifically overridden.

Ctrl code	Action
Ctrl-A	Position to end (END)
Ctrl-B	Position left word
Ctrl-E	Insert Mode ON/OFF
Ctrl-F	Position right word
Ctrl-I	Destructive tab (TAB)
Ctrl-J	(Newline) Accepts the entire field (ESCAPE 00)
Ctrl-M	(Carriage Return) Acts like a Newline except on DG terminals where it is a truncating terminator (ESCAPE 00)
Ctrl-N	Position back tab
Ctrl-O	Position forward tab
Ctrl-P	Position to beginning (HOME)
Ctrl-R	Delete a character
Ctrl-T	Backspace
Ctrl-U	Refresh screen,
Ctrl-V	Erase to end of field
ESC (Ctrl-[ )	Sends an ESC (ESCAPE 01)

Decimal code 32 (space) - 126 (~) along with their 8-bit counterparts 160 - 254 are passed through as valid characters to the COBOL programs.

Function keys (other than ESC) are treated just as DG function keys in that they accept the entire field and terminate the ACCEPT with ESCAPE KEY set to the appropriate value. These will be defined as ESCAPE *nn* where *nn* will refer to the ESCAPE KEY returned to the COBOL program.

The following is a generalization for the keycap legends, if they exist; not all keyboards will generate a Shift state for each key.

code	action
TAB	Destructive tab
Shift-TAB	Position back tab
Ins,Insert	Insert Mode ON/OFF
HOME	Position to beginning

END	Position to end
Shift-END	Position to beginning
DEL	Backspace (**WARNING** this may be the Linux Intr key)
↑	Move to previous field (Beep at top)
↓	Move to next field (Fall out at bottom)
→	Move right a character (Beep at end of field)
←	Move left a character (Beep at beginning of field)
Shift-↑	ESCAPE 70
Shift-↓	ESCAPE 77
Shift-→	ESCAPE 71
Shift-←	ESCAPE 72

The following keys are generally ignored by Interactive COBOL unless otherwise configured.

PageUp PageDown Page Send ..

For the best support with all the DG function keys, a terminal that supports the DG emulation is preferred; otherwise not all function keys with their appropriate shift states may be available.

To be able to run on various terminal types, the use of function keys should be restricted to the lowest common denominator (i.e., the one with the fewest function keys). There are not many terminals that support the Ctrl-Shift state of function keys. There are also not many terminals that have more than 10 or 12 function keys. A good start is to use only function keys f1 - f10 in base and shifted states before moving on to additional keys.

In the following tables when the generation of a key is described, the backslash character (\) implies that the next three digits comprise an octal code returned by the key.

B. DG

DG type (ICTERM=dg)

Assumes a DG D200 or upward compatible terminal.

Color is supported on D220, D230, and D470 type-terminals.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

legend	action
ERASE EOL (Ctrl-K)	Erase to end
ERASE PAGE (Ctrl-L)	Erase entire field
Shift-CMD-PRINT (\036\001)	ESCAPE 74
Shift-HOME (\036\010)	ESCAPE 75
CMD-PRINT (\036\021)	ESCAPE 73

Function Keys

	ESCAPE KEY nn values				Generated by \036 (plus the following)			
	Normal	Shift	Ctrl	Ctrl-Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1	2	10	18	26	q	a	1	!
F2	3	11	19	27	r	b	2	"
F3	4	12	20	28	s	c	3	#
F4	5	13	21	29	t	d	4	$
F5	6	14	22	30	u	e	5	%
F6	7	15	23	31	v	f	6	&
F7	8	16	24	32	w	g	7	'
F8	9	17	25	33	x	h	8	(
F9	34	41	48	55	y	i	9	)
F10	35	42	49	56	z	j	:	*
F11	36	43	50	57	{	k	;	+
F12	37	44	51	58	|	l	<	,
F13	38	45	52	59	}	m	=	-
F14	39	46	53	60	~	n	>	.
F15	40	47	54	61	p	`	0	/
C1	62	66			\	X		
C2	63	67			]	Y		
C3	64	68			^	Z		
C4	65	69			_	[		
↑		70				\027		
↓		77				\032		
→		71				\030		
←		72				\031		

Notes:

1. When using a non-DG terminal in a DG emulation mode, make sure the terminal does NOT have transmit XON/XOFF enabled since ^S and ^Q are used as real control codes when going from the host system to the terminal.

2. When positioning past column 126 on a DG terminal (or emulation), ICRUN will use the D400 window positioning code since the Ctrl-P sequence is not valid for these locations.

3. For DG terminals that support an ANSI mode, it may be advisable to run the terminal in the ANSI mode while under the Linux shell but switch to DG mode when running any COBOL programs (especially those that use function keys). To do this you can build a script file with the appropriate editor that will switch the terminal from ANSI mode to DG mode, execute ICRUN, and when ICRUN terminates switch back to ANSI mode. An example of this is shown below for a DG D215 terminal.

   To switch to DG mode from ANSI mode the characters \033[<3l must be sent.
   To switch to ANSI mode from DG mode the characters \036F@ must be sent.

   Thus an example execution file could be:

   ```
   echo "\033[<3l"
   icrun
   echo "\036F@"
   ```

   to switch from ANSI to DG mode, execute ICRUN, and return to ANSI mode.

   D210, 211, 214, 215, 400, 410, 411, 450, and 460 terminals support the above sequence.

   On other terminals you should use the appropriate command or escape sequences to perform these actions.

C. DGUNIX

DGUNIX type (ICTERM=dgunix)

Assumes a DG D217+ or upward compatible terminal.

DGUNIX is very similar to DG but no binary key sequences or command sequences are used. All sequences are in ASCII.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

legend	action
DEL (\010)	Backspace
ERASE EOL (\036PE)	Erase to end
ERASE PAGE (\036PH)	Erase entire field
HOME (\036PF)	Position to beginning
Shift-CMD-PRINT (\036P1)	ESCAPE 74
Shift-HOME (\036Pf)	ESCAPE 75
CMD-PRINT (\036P0)	ESCAPE 73

Function Keys

	ESCAPE KEY nn values				Generated by \036 (plus the following)			
	Normal	Shift	Ctrl	Ctrl-Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1	2	10	18	26	q	a	1	!
F2	3	11	19	27	r	b	2	"
F3	4	12	20	28	s	c	3	#
F4	5	13	21	29	t	d	4	$
F5	6	14	22	30	u	e	5	%
F6	7	15	23	31	v	f	6	&
F7	8	16	24	32	w	g	7	'
F8	9	17	25	33	x	h	8	(
F9	34	41	48	55	y	i	9	)
F10	35	42	49	56	z	j	:	*
F11	36	43	50	57	{	k	;	+
F12	37	44	51	58	|	l	<	,
F13	38	45	52	59	}	m	=	-
F14	39	46	53	60	~	n	>	.
F15	40	47	54	61	p	`	0	/
C1	62	66			\	X		
C2	63	67			]	Y		
C3	64	68			^	Z		
C4	65	69			_	[		
↑		70			PA	Pa		
↓		77			PB	Pb		
→		71			PC	Pc		
←		72			PD	Pd		

D. **AIX**

AIX type (ICTERM=aix)

Assumes the master console (high function terminal) under AIX.

Additional Input Keys:

Function Keys

	ESCAPE KEY nn values				Generated by \033[(plus the following)			
	Normal	Shift	Ctrl	Alt	Normal	Shift	Ctrl	Alt
F1	2	10	18	26	001q	013q	025q	037q
F2	3	11	19	27	002q	014q	026q	038q
F3	4	12	20	28	003q	015q	027q	039q
F4	5	13	21	29	004q	016q	028q	040q
F5	6	14	22	30	005q	017q	029q	041q
F6	7	15	23	31	006q	018q	030q	042q
F7	8	16	24	32	007q	019q	031q	043q
F8	9	17	25	33	008q	020q	032q	044q
F9	34	41	48	55	009q	021q	033q	045q
F10	35	42	49	56	010q	022q	034q	046q
F11	36	43	50	57	011q	023q	035q	047q
F12	37	44	51	58	012q	024q	036q	048q

E. ANSI

Ansi type (ICTERM=ansi)

Assumes a standard ANSI or upward compatible terminal.

Color is supported if the corresponding terminal supports the ANSI color sequences of:

ESC [3*f*;4*b*m

where

f is the foreground color (0 - 7)
b is the background color (0 - 7).

colors are Ascii:
0 - black, 1 - red, 2 - green, 3 - yellow, 4 - blue, 5 - magenta, 6 - cyan, 7 - white.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

legend	action	Generated by
PageUp	ESCAPE 63	\033[5~
PageDown	ESCAPE 65	\033[6~

Function Keys

	ESCAPE KEY nn values	Generated by
	Normal	Normal
F1	2	\033OP
F2	3	\033OQ
F3	4	\033OR
F4	5	\033OS
F5	6	\033[15~
F6	7	\033[17~
F7	8	\033[18~
F8	9	\033[19~
F9	34	\033[20~
F10	35	\033[21~
F11	36	\033[23~
F12	37	\033[24~

The above values are what the Windows XP telnet client provides. (added in 3.44)

F. ATT

AT&T type (ICTERM=att)

Assumes an AT&T 605 or upward compatible terminal.

Color is supported if the corresponding terminal supports the ANSI color sequences.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

legend	action
END	Erase to end
Shift <-	ESCAPE 72
Shift ->	ESCAPE 71
Shift-Insert	Insert ON/OFF
Insert	Insert ON/OFF
Shift-Up-arrow	ESCAPE 70
Shift-Down-arrow	Down-arrow

Function Keys

ESCAPE KEY nn values		Generated by \033 (plus the following)	
Normal	Shift	Normal	Shift
F1 2	10	Oc	OC
F2 3	11	Od	OD
F3 4	12	Oe	OE
F4 5	13	Of	OF
F5 6	14	Og	OG
F6 7	15	Oh	OH
F7 8	16	Oi	OI
F8 9	17	Oj	OJ
F9 34	41	No	NO
F10 35	42	Np	NP
F11 36	43	Nq	NQ
F12 37	44	Nr	NR
F13 38	45	Ns	NS
F14 39	46	Nt	NT

G. FILE

File type (ICTERM=file)

Assumes a standard 8-bit ASCII format with no control codes generated for positioning except for carriage-return, line-feed, form-feed, and spaces. No function keys are supported in this entry.

This is most useful as a detached job.

Additional Input Keys:

None.

H. FREEDOM

Freedom type (ICTERM=freedom)

Assumes a Freedom ONE type terminal.

No compressed mode support.

Additional Input Keys:

Function Keys

ESCAPE KEY nn values		Generated by Ctrl-A (followed by below) then <cr>	
Normal	Shift	Normal	Shift
F1 2	10	\040	\141
F2 3	11	A	a
F3 4	12	B	b
F4 5	13	C	c
F5 6	14	D	d
F6 7	15	E	e
F7 8	16	F	f
F8 9	17	G	g
F9 34	41	H	h
F10 35	42	I	i
F11 36	43	J	j
F12 37	44	K	k
F13 38	45	L	l
F14 39	46	M	m
F15 40	47	N	n

I. IBM

IBM type (ICTERM=ibm)

Assumes an IBM 3101 or upward compatible terminal with the turnaround character set to carriage-return (\015).

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

legend	action
ER INP	Erase field
CLEAR	Erase Field
ERASE EOF	Erase to end
PRINT	ESCAPE 73

Function Keys

ESCAPE KEY nn values — Generated by \033 (plus the following)

	Normal	Shift	Ctrl-Shift	Normal	Shift	Ctrl-Shift
F1	2	10	26	a\015	!a\015	"a\015
F2	3	11	27	b\015	!b\015	"b\015
F3	4	12	28	c\015	!c\015	"c\015
F4	5	13	29	d\015	!d\015	"d\015
F5	6	14	30	e\015	!e\015	"e\015
F6	7	15	31	f\015	!f\015	"f\015
F7	8	16	32	g\015	!g\015	"g\015
F8	9	17	33	h\015	!h\015	"h\015
F9	34	41	55	i\015	!i\015	"i\015
F10	35	42	56	j\015	!j\015	"j\015
F11	36	43	57	k\015	!k\015	"k\015
F12	37	44	58	l\015	!l\015	"l\015

PA1, PA2, PA3, Del LN, Ins LN, Jump, Print Line, Pr Msg, Send.. are all Ignored.

To support character attributes, more than a 3101 terminal must be available.

J. LINUX

LINUX type (ICTERM=linux)

Assumes the master console under Linux.

Color is supported if the corresponding monitor/emulator/terminal supports the ANSI color sequences.

A file is included in the examples directory, called linuxadd.map, that increases the function key support on the Linux master console to Shift-F9 - Shift-F12, Ctrl-F1 - Ctrl-F12, and Ctrl-Shift F1 - Ctrl-Shift F12. This file is used as "loadkeys linuxadd.map".

Please see the Linux documentation on loadkeys/dumpkeys for more information.

Additional Input Keys:

Function Keys

	ESCAPE KEY nn values				Generated by \033[(plus the following)			
	Normal	Shift	Ctrl	Ctrl-Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1	2	10	18	26	[A	25~	39~	51~
F2	3	11	19	27	[B	26~	40~	52~
F3	4	12	20	28	[C	28~	41~	53~
F4	5	13	21	29	[D	29~	42~	54~
F5	6	14	22	30	[E	31~	43~	55~
F6	7	15	23	31	17~	32~	44~	56~
F7	8	16	24	32	18~	33~	45~	57~
F8	9	17	25	33	19~	34~	46~	58~
F9	34	41	48	55	20~	35~	47~	59~
F10	35	42	49	56	21~	36~	48~	60~
F11	36	43	50	57	23~	37~	49~	61~
F12	37	44	51	58	24~	38~	50~	62~

K. MAC

Ansi type (ICTERM=mac)

Assumes a default Terminal session on an Apple Mac OS X console

Color is supported.

ESC [3*f*;4*b*m

where

f is the foreground color (0 - 7)
b is the background color (0 - 7).

colors are Ascii:
0 - black, 1 - red, 2 - green, 3 - yellow, 4 - blue, 5 - magenta, 6 - cyan, 7 - white.

Additional Input Keys:

legend	action	Generated by
PageUp	ESCAPE 63	\033[5~
PageDown	ESCAPE 65	\033[6~

Function Keys

	ESCAPE KEY nn values	Generated by
	Normal	Normal
F1	2	\033OP
F2	3	\033OQ
F3	4	\033OR
F4	5	\033OS
F5	6	\033[15~
F6	7	\033[17~
F7	8	\033[18~
F8	9	\033[19~
F9	34	\033[20~
F10	35	\033[21~
F11	36	\033[23~
F12	37	\033[24~
Shift F1	10	\033Op
Shift F2	11	\033Oq
Shift F3	12	\033Or
Shift F4	13	\033Os
Shift F5	14	\033[25~
Shift F6	15	\033[26~
Shift F7	16	\033[28~
Shift F8	17	\033[29~
Shift F9	44	\033[31~
Shift F10	45	\033[22~
Shift F11	46	\033[33~
Shift F12	47	\033[34~

Note: To use F9-F12 you may need to remove them from use by Dashboard.

L. PCWINDOW

PCWINDOW type (ICTERM=pcwindow, pcwindowcolor)

This entry is only available when running **ICOBOL** on Windows.

This entry only makes sense when running on the Master Console.

This entry uses different colors for attributes.

Color is supported if the corresponding monitor supports color and/or shades.

At startup, the ICLINES and/or ICCOLUMNS settings will be used to set the display to have the given number of rows and columns. If running in full-screen mode make sure that the values work with the native graphics card. At termination, the screen is returned to its original setting.

Compressed mode is supported by setting ICCOLUMNS to min:max.

Additional Input Keys:

legend	action
PageUp	ESCAPE 63
PageDown	ESCAPE 65
Ctrl-PageUp	ESCAPE 67
Ctrl-PageDown	ESCAPE 69

Function Keys

ESCAPE KEY nn values

	Normal	Shift	Ctrl	Alt or Ctrl-Shift
F1	2	10	18	26
F2	3	11	19	27
F3	4	12	20	28
F4	5	13	21	29
F5	6	14	22	30
F6	7	15	23	31
F7	8	16	24	32
F8	9	17	25	33
F9	34	41	48	55
F10	35	42	49	56
F11	36	43	50	57
F12	37	44	51	58

↑ (Uparrow)	70
↓ (Downarrow)	77
→ (Rightarrow)	71
← (Leftarrow)	72

Use ICCONFIG's or ICEDCFW's Terminal Description menu to print a listing of all the supported keys.

M. PCWINDOWMONO

PCWINDOW type (ICTERM=pcwindowmono)

This entry is only available when running **ICOBOL** on Windows and is only useful when running on the Master Console.

This entry is just like pcwindow but it is monochrome only. No attributes are mapped to colors. Colors are supported only with actual Color combinations. The default is White on black.

Two ways to have a different color other than White-on-Black

A different color combination can easily be chosen by simply replacing the particular color that you wish to change in the tdi file. For example, if you want green on black just go into the Configure Color/Attribute Map and change the "White" selections to "Green" leaving everything else the same. Save to the same or a separate name like "pcwindowgreen".

or

Use the environment variables ICCOLOR=process, ICFGCOLOR=green. This will set green on black. Or also set ICBGCOLOR=white and you will have Green-on-White.

At startup, the ICLINES and/or ICCOLUMNS settings will be used to set the display to have the given number of rows and columns. If running in full-screen mode make sure that the values work with the native graphics card. At termination, the screen is returned to its original setting.

Compressed mode is supported by setting ICCOLUMNS to min:max.

Additional Input Keys:

legend	action
PageUp	ESCAPE 63
PageDown	ESCAPE 65
Ctrl-PageUp	ESCAPE 67
Ctrl-PageDown	ESCAPE 69

Function Keys

ESCAPE KEY nn values

	Normal	Shift	Ctrl	Alt or Ctrl-Shift
F1	2	10	18	26
F2	3	11	19	27
F3	4	12	20	28
F4	5	13	21	29
F5	6	14	22	30
F6	7	15	23	31
F7	8	16	24	32
F8	9	17	25	33
F9	34	41	48	55
F10	35	42	49	56
F11	36	43	50	57
F12	37	44	51	58

↑ (Uparrow)	70
↓ (Downarrow)	77
→ (Rightarrow)	71
← (Leftarrow)	72

Use ICCONFIG's or ICEDCFW's Terminal Description menu to print a listing of all the supported keys.

N. SUN

SUN type (ICTERM=sun)

Assumes the master console on a SUN workstation.

No compressed mode support.

Additional Input Keys:

Function Keys

ESCAPE KEY nn values		Generated by	
	Normal		Normal
F1	2		\033[224z
F2	3		\033[225z
F3	4		\033[226z
F4	5		\033[227z
F5	6		\033[228z
F6	7		\033[229z
F7	8		\033[230z
F8	9		\033[231z
F9	34		\033[232z
F10	35		\033[233z
F11	36		\033[234z
F12	37		\033[235z

O. TERMINFO

TERMINFO type (ICTERM=terminfo)

This entry is only available when running on Linux.

Uses the Linux terminfo entry for the terminal specified by the TERM environment variable.

By default, carriage-return and new-line both accept the entire field, i.e., no truncation of the field.

No compressed mode support.

Additional Input Keys:

Terminfo Variable	Terminfo Capname	**ICOBOL** action
key_a1	ka1	C1 (ESCAPE 62)
key_a3	ka3	C2 (ESCAPE 63)
key_backspace	kbs	Backspace
key_btab	kcbt	Position back tab
key_c1	kc1	C3 (ESCAPE 64)
key_c3	kc3	C4 (ESCAPE 65)
key_clear	kclr	Erase entire field
key_dc	kdch1	Delete character
key_down	kcud1	Move to next field
key_end	kend	Position to end
key_enter	kent	Accepts the entire field
key_eol	kel	Erase to end
key_eos	ked	Erase entire field
key_f0	kf0	Accepts the entire field
key_f1 - key_f15	kf1 - kf15	normal F1 - F15
key_f16 - key_f30	kf16 - kf30	Shift-(F1 - F15)
key_f31 - key_f45	kf31 - kf45	Ctrl-(F1 - F15)
key_f46 - key_f60	kf46 - kf60	Ctrl-Shift-(F1 - F15)
key_f61 - key_f63	kf61 - kf63	Not currently used by Interactive COBOL
key_home	khome	Position to beginning
key_ic, key_il	kich1, kil1	Insert Mode ON/OFF
key_left	kcub1	Move left a character
key_print	kprt	CMD-Print (ESCAPE 73)
key_right	kcuf1	Move right a character
key_sdc	kDC	Delete character
key_send	kEND	Position to beginning
key_shome	kHOM	Position to end
key_sleft	kLFT	Shift Left-arrow (ESCAPE 72)
key_sprint	kPRT	Shift CMD-Print (ESCAPE 74)
key_sright	kRIT	Shift Right-arrow (ESCAPE 71)
key_stab	khts	Position back tab
key_sr	kri	Shift Up-arrow (ESCAPE 70)
key_up	kcuu1	Move to previous field
key_sf	kind	Shift Down-arrow (ESCAPE 77)

When defining the functions keys, (key_f0 - key_f65), there must not be a hole, i.e., a particular function key cannot be skipped, or else any program using the Linux curses interface (not Interactive COBOL) will not see any function key after the hole.

If the ICLINES and/or ICCOLUMNS are not set, then the LINES and/or COLUMNS given by terminfo is used. If ICLINES and/or ICCOLUMNS are given, they override the terminfo settings.

Terminfo is the Linux terminal information database. More can be found on terminfo by looking in the appropriate Linux manual or using the Linux *man* command with terminfo as the argument.

For Output:

Interactive COBOL will use the following terminfo attributes:

Normal (no attributes set),
Underline (p2),
Reverse (p3),
Blink (p4), and
Bright (or Bold)(p6).

These attributes should be defined in the terminfo definition with the sgr entry. Other definitions in the terminfo database that Interactive COBOL will use are bell (bel), clear_screen (clear), clr_eol (el), clr_eos (ed), cursor_address (cup), cursor_left (cub1), cursor_right (cuf1), delete_character (dch1), print_screen (mc0), prtr_off (mc4), prtr_on (mc5), and scroll_forward (ind).

The following attributes are not currently used by Interactive COBOL, standout (p1), dim (p5), invisible (p7), protected (p8), or alternate character set (p9).

If the terminfo magic_cookie_glitch (xmc) entry is positive, all attribute information is ignored.

To help check and/or define a terminfo definition, the program termi.c has been included in the examples directory to show some simple uses of the terminfo database and how Interactive COBOL uses it. Both the source and executable are provided. The Linux *tputs* utility can be used as well. The Linux *infocmp* utility should be used to compare or print out a terminfo entry.

When looking up a keystroke (string), Interactive COBOL uses the first match in the table as seen under ICCONFIG.

When using ICCONFIG for the Terminfo base selection to Configure the Keyboard, the Terminfo Capname's can be entered by entering a backslash (\) followed by the appropriate Capname.

P. VT100

VT100 type (ICTERM=vt100)

Assumes a standard DEC VT100 or upward compatible terminal. Only four function keys are supported in this entry.

Color is supported if the corresponding terminal supports the ANSI color sequences.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

Function Keys

	ESCAPE KEY nn values	Generated by
	Normal	Normal
PF1	2	\033OP
PF2	3	\033OQ
PF3	4	\033OR
PF4	5	\033OS

Q. **VT220**

VT220 type (ICTERM=vt220)

Assumes a standard DEC VT220 or upward compatible terminal.

Color is supported if the corresponding terminal supports the ANSI color sequences.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

Function Keys

ESCAPE KEY	nn values Normal	Generated by Normal
PF1	2	\033OP
PF2	3	\033OQ
PF3	4	\033OR
PF4	5	\033OS

Function keys F6 - F20 are treated like F1 - F15

ESCAPE KEY	nn values Normal	Generated by \033 (plus the following) Normal
F6	2	[17~
F7	3	[18~
F8	4	[19~
F9	5	[20~
F10	6	[21~
F11	7	[23~
F12	8	[24~
F13	9	[25~
F14	34	[26~
F15	35	[28~
F16	36	[29~
F17	37	[31~
F18	38	[32~
F19	39	[33~
F20	40	[34~

R. VT220PC

VT220 type (ICTERM=vt220pc)

Assumes a standard Windows PC using an VT220 or Ansi emulator. Later Xterm emulation under Linux more closely match this emulation.

Color is supported if the corresponding terminal supports the ANSI color sequences.

Compressed mode is supported for terminals that support compressed mode.

Additional Input Keys:

Function Keys

ESCAPE KEY nn values					Generated by \033 (plus the following)			
	Normal	Shift	Ctrl	Ctrl Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1	2	10	18	26	[11~	[11;2~	[11:5~	[11:6~
F2	3	11	19	27	[12~	[12;2~	[12:5~	[12:6~
F3	4	12	20	28	[13~	[13;2~	[13:5~	[13:6~
F4	5	13	21	29	[14~	[13;2~	[14:5~	[14:6~
F5	6	14	22	30	[15~	[15;2~	[15:5~	[15:6~
F6	7	15	23	31	[17~	[17;2~	[17:5~	[17:6~
F7	8	16	24	32	[18~	[18;2~	[18:5~	[18:6~
F8	9	17	25	33	[19~	[19;2~	[19:5~	[19:6~
F9	34	41	48	55	[20~	[20;2~	[20:5~	[20:6~
F10	35	42	49	56	[21~	[21;2~	[21:5~	[21:6~
F11	36	43	50	57	[23~	[23;2~	[23:5~	[23:6~
F12	37	44	51	58	[24~	[24;2~	[24:5~	[24:6~

ESCAPE KEY nn values	Normal	Generated by Normal
PF1	2	\033OP
PF2	3	\033OQ
PF3	4	\033OR
PF4	5	\033OS

S. WYSE and WY50

WYSE type (ICTERM=wyse)

Assumes a WYSE 60 or upward compatible terminal with hidden attributes.

Compressed mode is supported for terminals that support compressed mode.

WYSE50 type (ICTERM=wy50)

Assumes a WYSE 50 terminal with non-hidden attributes. In this case, only a single attribute is supported. Any Blink, Bright, Reverse, or Underline attribute will use write-protect mode which can be configured on a Wyse-50 to either dim, normal, or reverse. Thus you will be able to see two attributes (normal and protected) as long as the protect mode display attribute is NOT set to normal. We recommend that reverse be used, especially if the Screen Handler is also being used.

Write protect mode on and off are ESC ")" and ESC "(" respectively. This mode does not actual write protect the screen unless protect mode has been enabled with an ESC "&". Protect mode disable is ESC "'" (single-close-quote) and is the default.

Additional Input Keys:

legend	action
CLR LINE, ERASE EOF	Erase to end
CLR SCRN, ERASE EOP	Erase field
PRINT, Prt SC	ESCAPE 73

Function Keys

ESCAPE KEY nn values Generated by \001 (plus the following)

	Normal	Shift	Ctrl	Ctrl Shift	Normal	Shift	Ctrl	Ctrl Shift
F1	2	10	18	26	@\015	`\015	\200\015	\220\015
F2	3	11	19	27	A\015	a\015	\201\015	\221\015
F3	4	12	20	28	B\015	b\015	\202\015	\222\015
F4	5	13	21	29	C\015	c\015	\203\015	\223\015
F5	6	14	22	30	D\015	d\015	\204\015	\224\015
F6	7	15	23	31	E\015	e\015	\205\015	\225\015
F7	8	16	24	32	F\015	f\015	\206\015	\226\015
F8	9	17	25	33	G\015	g\015	\207\015	\227\015
F9	34	41	48	55	H\015	h\015	\210\015	\230\015
F10	35	42	49	56	I\015	i\015	\211\015	\231\015
F11	36	43	50	57	J\015	j\015	\212\015	\232\015
F12	37	44	51	58	K\015	k\015	\213\015	\233\015
F13	38	45	52	59	L\015	l\015	\214\015	\234\015
F14	39	46	53	60	M\015	m\015	\215\015	\235\015
F15	40	47	54	61	N\015	n\015	\216\015	\236\015

F16 ignored ..

Next, Prev, Send, Send Line, SN Msg, Del Line, Ins Line are all Ignored.

For both WYSE and WY50

The down-arrow key sends a Ctrl-J which is normally a new-line, instead with ICTERM=wyse, Interactive COBOL treats a Ctrl-J like a normal down-arrow.

Given the behavior of the down-arrow, the Enter key (which sends a Ctrl-M (carriage-return) is treated like a normal new-line and acts as an Terminator with no Erase. (Thus, there is no key on a Wyse like the DG carriage-return, i.e., that functions as a Terminator with Erase.)

The backspace key and the left-arrow key send a Ctrl-H. By default, Interactive COBOL treats a Ctrl-H like a left-arrow

For the Ctrl and Ctrl-Shift function keys, the line that the terminal is on must be running in 8-bit mode and ISTRIP must be off since the high order bit is determining the function key.

T. XENIX and SCO

XENIX or SCO type (ICTERM=xenix or ICTERM=sco)

Assumes the master console under SCO UNIX.

Many terminal emulators for PC's support an SCO master console mode.

Some terminals have an SCO or XENIX or UNIX mode that matches this description.

Color is supported if the corresponding monitor/emulator/terminal supports the ANSI color sequences.

For SCO Unixware, the file at386.ini in the examples directory can be used with the mapstr command to load unique Ctrl- and Ctrl-Shift functions keys for the master console. See the man page on mapstr for more information.

Additional Input Keys:

Function Keys

ESCAPE KEY nn values				Generated by \033[(plus the following)			
Normal	Shift	Ctrl	Ctrl-Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1 2	10	18	26	M	Y	k	w
F2 3	11	19	27	N	Z	l	x
F3 4	12	20	28	O	a	m	y
F4 5	13	21	29	P	b	n	z
F5 6	14	22	30	Q	c	o	@
F6 7	15	23	31	R	d	p	[
F7 8	16	24	32	S	e	q	\
F8 9	17	25	33	T	f	r	]
F9 34	41	48	55	U	g	s	^
F10 35	42	49	56	V	h	t	_
F11 36	43	50	57	W	i	u	`(open)
F12 37	44	51	58	X	j	v	{

U. XTERM-FK

XTERM-FK type (ICTERM=xterm-fk)

Assumes an XWINDOW terminal emulation using these function keys.

Additional Input Keys:

Function Keys

ESCAPE KEY	nn values	Generated by
	Normal	Normal
F1	2	\033[11~
F2	3	\033[12~
F3	4	\033[13~
F4	5	\033[14~
F5	6	\033[15~
F6	7	\033[17~
F7	8	\033[18~
F8	9	\033[19~
F9	34	\033[20~
F10	35	\033[21~
F11	36	\033[23~
F12	37	\033[24~

V. 386ix and AT386

386/ix or AT386 type (ICTERM=386ix or ICTERM=at386)

Assumes the master console under Interactive UNIX (formerly 386/ix), standard AT&T UNIX, SCO Openserver 6.

Color is supported if the corresponding terminal supports the ANSI color sequences.

Additional Input Keys:

Function Keys

ESCAPE KEY nn values					Generated by			
	Normal	Shift	Ctrl	Ctrl Shift	Normal	Shift	Ctrl	Ctrl-Shift
F1	2	10	18	26	\033OP	\033Op	\033OB	\033Ob
F2	3	11	19	27	\033OQ	\033Oq	\033OC	\033Oc
F3	4	12	20	28	\033OR	\033Or	\033OD	\033Od
F4	5	13	21	29	\033OS	\033Os	\033OE	\033Oe
extended vt100 codes								
F5	6	14	22	30	\033OT	\033Ot	\033OF	\033Of
F6	7	15	23	31	\033OU	\033Ou	\033OG	\033Og
F7	8	16	24	32	\033OV	\033Ov	\033OH	\033Oh
F8	9	17	25	33	\033OW	\033Ow	\033OI	\033Oi
F9	34	41	48	55	\033OX	\033Ox	\033OJ	\033Oj
F10	35	42	49	56	\033OY	\033Oy	\033OK	\033Ok
F11	36	43	50	57	\033OZ	\033Oz	\033OL	\033Ol
F12	37	44	51	58	\033OA	\033Oa	\033OM	\033Om

The Ctrl and Ctrl-Shift states of the function keys can be set to the above values by running the I/UNIX ttymap command on a mapfile that adds those keys to the default keyboard mapping. (The default maps Ctrl-F1 and F1 to the same value for example.)

A file (at386.map) has been included in the Linux release under the examples directory that provides a sample mapfile that can be used for this purpose.

Note: This addition can be used in previous revisions of Interactive COBOL by adding the new function keys using ICCONFIG to the 386at terminal description file.

Under SCO, the Ctrl and Ctrl-Shift states must be set with the mapkey utility.

APPENDICES

APPENDIX A. ASCII CODES

Dec	Oct	Hex	DG Function	Ctrl-code	PC Function/Character
0	000	00	Null	Ctrl @	NUL space
1	001	01	Print Screen Form	Ctrl A	SOH ☺
2	002	02	Reverse off	Ctrl B	STX ☻
3	003	03		Ctrl C	ETX ♥
4	004	04		Ctrl D	EOT ♦
5	005	05	Read cursor address	Ctrl E	ENQ ♣
6	006	06		Ctrl F	Ack ♠
7	007	07	Bell	Ctrl G	Bell •
8	010	08	Cursor Home	Ctrl H	Backspace ◘
9	011	09		Ctrl I	HTab ○
10	012	0A	Newline	Ctrl J	Linefeed ◙
11	013	0B	Erase EOL	Ctrl K	VTab ♂
12	014	0C	Erase Screen	Ctrl L	Form-feed ♀
13	015	0D	Carriage Return	Ctrl M	Carriage Return ♪
14	016	0E	Blink ON	Ctrl N	SO ♫
15	017	0F	Blink off	Ctrl O	SI ☼
16	020	10	Write cursor addr(c,r)	Ctrl P	DLE ►
17	021	11	Print Screen	Ctrl Q	DC1 (XON) ◄
18	022	12	Roll Enable	Ctrl R	DC2 ↕
19	023	13	Roll Disable	Ctrl S	DC3 (XOFF) ‼
20	024	14	Underscore ON	Ctrl T	DC4 ¶
21	025	15	Underscore OFF	Ctrl U	NAK §
22	026	16	Reverse On	Ctrl V	SYN ▬
23	027	17	Cursor Up	Ctrl W	ETB ↨
24	030	18	Cursor Right	Ctrl X	CAN ↑
25	031	19	Cursor Left	Ctrl Y	EM ↓
26	032	1A	Cursor Down	Ctrl Z	SUB →
27	033	1B	Escape	Ctrl [	ESC ←
28	034	1C	Dim ON	Ctrl \	FS ∟
29	035	1D	Dim OFF	Ctrl]	GS ↔
30	036	1E	Command Header	Ctrl ^	RS ▲
31	037	1F		Ctrl _	US ▼

Dec	Oct	Hex	DG	PC	Dec	Oct	Hex	DG	PC	Dec	Oct	Hex	DG	PC
32	040	20	space	space	64	100	40	@	@	96	140	60	`	`
33	041	21	!	!	65	101	41	A	A	97	141	61	a	a
34	042	22	"	"	66	102	42	B	B	98	142	62	b	b
35	043	23	#	#	67	103	43	C	C	99	143	63	c	c
36	044	24	$	$	68	104	44	D	D	100	144	64	d	d
37	045	25	%	%	69	105	45	E	E	101	145	65	e	e
38	046	26	&	&	70	106	46	F	F	102	146	66	f	f
39	047	27	’	’	71	107	47	G	G	103	147	67	g	g
40	050	28	(	(	72	110	48	H	H	104	150	68	h	h
41	051	29	)	)	73	111	49	I	I	105	151	69	i	i
42	052	2A	*	*	74	112	4A	J	J	106	152	6A	j	j
43	053	2B	+	+	75	113	4B	K	K	107	153	6B	k	k
44	054	2C	,	(comma),	76	114	4C	L	L	108	154	6C	l	l
45	055	2D	-	-	77	115	4D	M	M	109	155	6D	m	m
46	056	2E	.	.	78	116	4E	N	N	110	156	6E	n	n
47	057	2F	/	/	79	117	4F	O	O	111	157	6F	o	o
48	060	30	0	0	80	120	50	P	P	112	160	70	p	p
49	061	31	1	1	81	121	51	Q	Q	113	161	71	q	q
50	062	32	2	2	82	122	52	R	R	114	162	72	r	r
51	063	33	3	3	83	123	53	S	S	115	163	73	s	s
52	064	34	4	4	84	124	54	T	T	116	164	74	t	t
53	065	35	5	5	85	125	55	U	U	117	165	75	u	u
54	066	36	6	6	86	126	56	V	V	118	166	76	v	v
55	067	37	7	7	87	127	57	W	W	119	167	77	w	w
56	070	38	8	8	88	130	58	X	X	120	170	78	x	x
57	071	39	9	9	89	131	59	Y	Y	121	171	79	y	y
58	072	3A	:	:	90	132	5A	Z	Z	122	172	7A	z	z
59	073	3B	;	;	91	133	5B	[	[	123	173	7B	{	{
60	074	3C	<	<	92	134	5C	\	\	124	174	7C	|	|
61	075	3D	=	=	93	135	5D	]	]	125	175	7D	}	}
62	076	3E	>	>	94	136	5E	^	^	126	176	7E	~	~
63	077	3F	?	?	95	137	5F	_	_	127	177	7F	DEL	⌂

Dec	Oct	Hex	DGI	PC
128	200	80		Ç
129	201	81		ü
130	202	82		é
131	203	83		â
132	204	84		ä
133	205	85		à
134	206	86		å
135	207	87		ç
136	210	88		ê
137	211	89		ë
138	212	8A		è
139	213	8B		ï
140	214	8C		î
141	215	8D		ì
142	216	8E		Ä
143	217	8F		Å
144	220	90		É
145	221	91		æ
146	222	92		Æ
147	223	93		ô
148	224	94		ö
149	225	95		ò
150	226	96		û
151	227	97		ù
152	230	98		ÿ
153	231	99		Ö
154	232	9A		Ü
155	233	9B		¢
156	234	9C		£
157	235	9D		¥
158	236	9E		₧
159	237	9F		ƒ
160	240	A0	space	á
161	241	A1	¬	í
162	242	A2	½	ó
163	243	A3	µ	ú
164	244	A4	²	ñ
165	245	A5	³	Ñ
166	246	A6	¤	ª
167	247	A7	¢	º
168	250	A8	£	¿
169	251	A9	ª	⌐
170	252	AA	º	¬
171	253	AB	¡	½
172	254	AC	¿	¼
173	255	AD	©	¡
174	256	AE	®	«
175	257	AF	‡	»
176	260	B0	»	░
177	261	B1	«	▒
178	262	B2	¶	▓
179	263	B3	™	│
180	264	B4	ƒ	┤
181	265	B5	¥	╡
182	266	B6	±	╢
183	267	B7	≤	╖
184	270	B8	≥	╕
185	271	B9	·	╣
186	272	BA	` (grave)	║
187	273	BB	§	╗
188	274	BC	° (degree)	╝
189	275	BD	¨ (umlaut)	╜
190	276	BE	´ (acute)	╛
191	277	BF	↑	┐

Dec	Oct	Hex	DGI	PC
192	300	C0	Á	└
193	301	C1	À	┴
194	302	C2	Â	┬
195	303	C3	Ä	├
196	304	C4	Ã	─
197	305	C5	Å	┼
198	306	C6	Æ	╞
199	307	C7	Ç	╟
200	310	C8	É	╚
201	311	C9	È	╔
202	312	CA	Ê	╩
203	313	CB	Ë	╦
204	314	CC	Í	╠
205	315	CD	Ì	═
206	316	CE	Î	╬
207	317	CF	Ï	╧
208	320	D0	Ñ	╨
209	321	D1	Ó	╤
210	322	D2	Ò	╥
211	323	D3	Ô	╙
212	324	D4	Ö	╘
213	325	D5	Õ	╒
214	326	D6	Ø	╓
215	327	D7	Œ	╫
216	330	D8	Ú	╪
217	331	D9	Ù	┘
218	332	DA	Û	┌
219	333	DB	Ü	▉
220	334	DC	space	▄
221	335	DD	Ÿ	▌
222	336	DE	space	▐
223	337	DF	space	▀
224	340	E0	á	α
225	341	E1	à	β
226	342	E2	â	Γ
227	343	E3	ä	π
228	344	E4	ã	Σ
229	345	E5	å	σ
230	346	E6	æ	µ
231	347	E7	ç	τ
232	350	E8	é	Φ
233	351	E9	è	Θ
234	352	EA	ê	Ω
235	353	EB	ë	δ
236	354	EC	í	∞
237	355	ED	ì	φ
238	356	EE	î	∈
239	357	EF	ï	∩
240	360	F0	ñ	≡
241	361	F1	ó	±
242	362	F2	ò	≥
243	363	F3	ô	≤
244	364	F4	ö	⌠
245	365	F5	õ	⌡
246	366	F6	ø	÷
247	367	F7	œ	≈
248	370	F8	ú	°
249	371	F9	ù	∙
250	372	FA	û	·
251	373	FB	ü	√
252	374	FC	β	ⁿ
253	375	FD	ÿ	²
254	376	FE	space	■
255	377	FF	space	space

▸ **Notes:**

1. Decimal codes 128 - 159 for DGI are the same as their 7-bit counterparts by default.
2. DGI is as defined by a D216E+/D217/D413/D463 terminal.

APPENDIX B. RS-232C

Two basic types of serial interfaces have been defined by the Electronic Industries Association (EIA). These are DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). These conventions specify the direction of information flow for data and control signals. The names of the signals are always based on their DTE function. Each position on a DTE connector performs a function which is complementary to the corresponding position on the DCE connector. This means that connections between DTE and DCE connectors can be made pin-for-pin. When making connections of the same type, the functions of the pins rather than the number of the pins, must be matched.

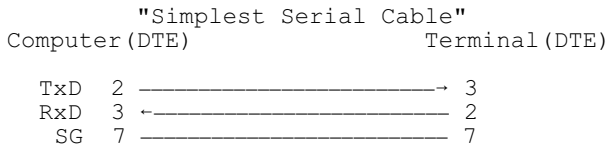
The connectors on terminals are almost all DTE, while the connectors on modems are almost all DCE. The connectors from computers can be either. The serial port(s) on the back of PC and AT class machines are almost always DTE. In the PC/AT world, almost all add-on serial boards (both dumb and smart) provide a DTE connection. On many Linux machines the connection is usually DCE. You should always check your equipment's documentation to determine if it is DTE or DCE.

To connect a DTE connection to a DCE connection using the same connector, only a straight-thru cable is needed. For example, to connect a terminal or a PC to a modem, a 25-pin straight-thru cable can be used.

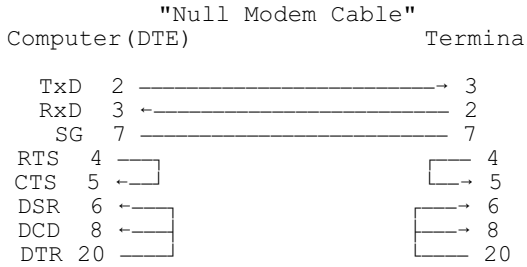
The following table lists the signals which are generally available on an RS-232C DTE DB25-pin or an AT DB9-pin connector. All the examples assume a DB25-pin connector.

Function	Signal	I/O	DB25	AT-DB9
Frame Ground	GND	-	Pin 1	plug
Transmit Data	TxD	O	Pin 2	Pin 3
Receive Data	RxD	I	Pin 3	Pin 2
Request To Send	RTS	O	Pin 4	Pin 7
Clear To Send	CTS	I	Pin 5	Pin 8
Data Set Ready	DSR	I	Pin 6	Pin 6
Signal Ground	GND	-	Pin 7	Pin 5
Data Carrier Detect	DCD	I	Pin 8	Pin 1
Data Terminal Ready	DTR	O	Pin 20	Pin 4
Ring Indicator	RI	I	Pin 22	Pin 9

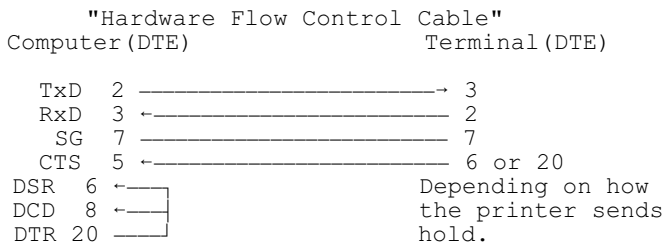
A typical RS-232C wiring diagram for standard hardwired terminals or serial printers not needing hardware flow control to a DTE connection can be made with the following 3-wire connection:



Some devices, especially older Data General terminals, require the 4-5 and 6-8-20 loop-back on the terminal side to go ON LINE, but most newer terminals have an option to disable this requirement. Read the documentation for the device to be connected to see if any of these loop-back schemes are required. Most systems do not require the 4-5 or 6-8-20 loop-back on the computer side.

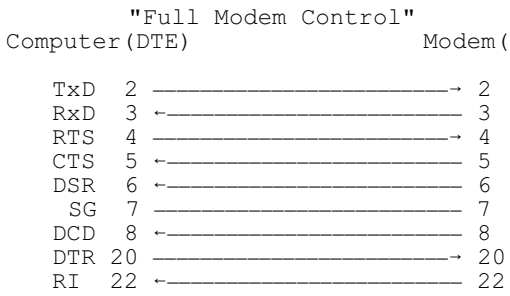


A typical wiring diagram for devices needing hardware flow control can be made with the following 4-wire connection:



Hardware flow control is used when a device (usually a printer) cannot send software flow control codes. The correct pin on the printer, usually 6 (DSR) or 20 (DTR), must be wired to pin 5 (CTS) such that the computer will know to stop sending data until pin 5 (CTS) is re-asserted.

A typical wiring diagram for devices needing modem control can be made with the following 9-wire connector:



In many cases, ring indicator (RI) is not needed or supported by the host device and in some cases the Data Set Ready (DSR) signal is not needed thus allowing an 8 or even 7-wire cable.

Hardware flow control is generally required for modems with data compression (MNP 5 or V.42bis) to keep the computer from overrunning its input buffer since hardware flow control is generally much faster than software flow control.

When using high speed modems with data compression support, the baud rate must be set to at least 56000 (for V.32bis modems) or 38400 (for V.32 modems) to realize the maximum throughput.

APPENDIX C. COMMON PROBLEMS On Linux

Following is a list of general problems that some users have encountered along with our initial suggestion on what to look at in your system.

1) When I start ICRUN the screen seems to be confused.

   Answer: Check your ICTERM entry. If set to terminfo, check the appropriate terminfo file. ICRUN makes a bigger demand on the terminfo settings, especially for function keys, then many other simpler applications. The sample termi program shipped with Interactive COBOL can also be used to help debug terminfo settings.

   If possible use a directly supported terminal instead of terminfo.

2) I cannot print from ICRUN.

   Answer: If you are using the Printer Control Utility or a printer control queue (@PCQ) make sure that from the Linux shell you can use the *lp* command with the destination given in the configuration file. If not, then either fix the configuration file or add that particular Linux print spooler.

If your problems still persist, fill out a Support Information Request (SIR) form using the one from the end of this manual as a guide or go online to www.icobol.com and submit a support request..

Send a copy of the ICINFO report for all reported problems.

For the fastest response to an Interactive COBOL problem, please follow these guidelines and if possible either E-mail or FAX. The E-mail address is support@icobol.com and the FAX number is (919) 851-4609.

INDEX