INGLENET
business solutions

TIP/ix Installation and Operations

IP-638

May 2016

This edition applies to TIP/ix 2.5 and revision levels of TIP/ix 2.5 until otherwise indicated in a new edition. Publications can be downloaded from the Inglenet website.

Inglenet Business Solutions Inc reserves the right to modify or revise this document without notice. Except where a Software Usage Agreement has been executed, no contractual obligation between Inglenet Business Solutions Inc and the recipient is either expressed or implied.

It is agreed and understood that the information contained herein is **Proprietary** and **Confidential** and that the recipient shall take all necessary precautions to ensure the confidentiality thereof.

If you have a license agreement for TIP Studio or TIP/ix with Inglenet Business Solutions Inc, you may make copies of this documentation for internal use. Otherwise, you may not copy or transmit this document, in whole or in part, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of Inglenet Business Solutions Inc.

Inglenet Business Solutions Inc

Toll Free: 1-800-387-9391

Website: http://www.inglenet.com
Sales: sales@inglenet.com
Help Desk: helpdesk@inglenet.com

TIP Studio, TIP/ix, and TIP/30, and are registered trademarks of Inglenet Business Solutions Inc:

This documentation occasionally makes reference to the products of other corporations. These product names may be trademarks, registered or otherwise, or service marks of these corporations. Where this is the case, they are hereby acknowledged as such by Inglenet Business Solutions Inc.

Contents

TIP/ix Installation and Operation

Introduction

What is TIP/ix

TIP/ix is an Online Transaction Processing system for the UNIX/LINUX operating system.

This document describes:

- the programming languages supported by TIP/ix,
- how to install TIP/ix,
- and how to use it to its full capability.

TIP/ix supports many different application program interfaces including TIP/30, IMS/90, SFS/90. Unisys OS2200 customers can use TIP/ix to run COBOL IMS/1100 applications as well as DPS/2200, TIP/HVTIP and DMS/2200

Inglenet has verified COBOL transactions with the Micro Focus COBOL compiler. See "TIP/ix Support for COBOL Compilers" in the ***TIP/ix Programming Reference*** for details.

TIP/ix also works with OpenCOBOL 1.1, OpenCOBOL 2.0, GNU Cobol 2.1 and COBOL-IT Enterprise Edition.

What is Supported

TIP/ix supports:

- TIP/ix transaction programs
- TIP/30 transaction programs
- IMS/90 transaction programs, if they use SFS for screen format processing and if the SFS screens can be converted successfully into TIP/30 MCS screens. A utility called SFSCNV, provided with TIP/30, converts SFS formats into native TIP MCS formats.
- IMS/90 and IMS/1100 transaction programs that use raw UNISCOPE messages with DICE codes and FCCs.
- ISAM files via D-ISAM from Byte Designs imbedded in TIP/ix.
- DPS/2200, IMS/2200, OS2200 TIP/HVTIP
- WebTS 2200
- UniAccess ODBC interface
- DMS/2200 applications supported, data is stored in Oracle (or SQL, MySQL) database

- RDMS/2200 can be converted to use embedded SQL and Oracle

What is Not Supported

TIP/ix currently does **not** support the following:

- The RPG programming language for transaction programs.
- IMS/90 programs, which use the Screen Management System (SMS), are not supported directly.

However, you can use the **SMSCNV** utility to convert SMS screens to SFS screens. The SMSCNV utility runs on the System 80 — not on UNIX.

- The subroutine call TIPCPAGE. The concept of a UNISCOPE terminal control page is not relevant in a UNIX environment.
- The subroutine TIPJUMP is not supported.
- By default, TIP/ix's IMS emulation normally sends the result of a CALL "BUILD" statement directly to the terminal (rather than placing it into your application's OMA).

To enable your IMS application to access (or modify) the results of the CALL BUILD in your OMA, use **smprog** to set "CALL BUILD XMIT" to "N".

The contents of the OMA might not be the same as with OS/3.

TIP/dbi

Any transaction that uses the DMS database on System/80 or the Unisys OS2200 systems will require the TIP Database Interface (TIP/dbi).

You can also use TIP/dbi to emulate indexed files with a supported database product. See the ***TIP/ix Release Notes*** for a list of supported database products.

The original version of TIP/dbi has been replaced by TIP/dbi II.

TIP/dbi II provides more flexibility and improved performance and supports Oracle via the Oracle Call Interface and possibly other databases such as Microsoft SQL Server via and ODBC interface and bridge software such as Easysoft.

TIP/ix Installation

Copy Software to UNIX

The TIP/ix software is distributed in UNIX "tar" format and is generally downloaded from the Inglenet web-site (www.inglenet.com). You may email helpdesk@inglenet.com to obtain a logon if you do not already have one. The file downloaded will have different names depending on the platform it is intended to be used on.

Be careful to copy or ftp in 'binary' mode. The tar file is not a simple text file.

Reinstall TIP/ix if already on 2.5 R0 – 0187 or later

If you already have TIP/ix installed and you are running version 2.5 R0 – 0187 or later there is a new way of installing an update of TIP/ix which does not require you to log in or su as the 'root' user or use sudo.

Copy the platform.tar file (e.g. REDHAT5.tar or AIX5.tar, etc) to some working directory (not '/'), then change (cd) to the directory and run from the Unix shell prompt:

```
tipctl install platform.tar
```

You need to be one of 'root', the TIP/ix admin user or a TIP/ix user who has 'master' level security to be allowed to run 'tipctl' to install an update.

If you want to install TIP/ix into a different location you can run:

```
tipctl install platform.tar /my/alt/tiproot
```

Where *platform.tar* is the TIP/ix release tar file and (if used) */my/alt/tiproot* is some other directory to install the update of TIP/ix into. By default this will install over top of the current TIP/ix system defined by the TIPROOT environment variable.

Be aware: this will shut the current TIP/ix system down if it is running.

If your Unix/Linux system has access to the Internet you can also have 'tipctl' do the download of the latest TIP/ix software by running:

tipctl download

When the download completes you can then use the 'tipctl install' command shown above.

You could also download and install by running:

tipctl update

Install TIP/ix new on UNIX/Linux

You must sign onto UNIX/Linux as "**root**" and make a temporary directory to hold this software. Depending on the system where you are installing TIP/ix, you may need from 60 megabytes to 100 megabytes of free hard disk memory. The UNIX **df** command (or equivalent) can be used to determine how much disk space is available.

If you have previously installed TIP/ix, backup your system before continuing with the installation of new software.

TIP/ix must not be running during this install/update procedure. Be certain to login to UNIX as the "root" user (do not "su" to root!).

Read the README file that comes with each update of TIP/ix. This file is contained in the tar file and will be copied to $TIPROOT/conf. It is also viewable on the Inglenet web site.

Example of TIP/ix Installation Dialogue:

```
mkdir /tmp/tipix
cd /tmp/tipix
tar xvf platform.tar
sh ./install
rm -rf /tmp/tipix
logout
```

The installation procedure involves:

1. Creating a temporary directory (with the suggested name "/tmp/tipix"). The TIP/ix release media will be copied to this directory.
2. Make the temporary directory the current directory.
3. Copy the release media into the temporary directory. The "tar" command options "xvf" are required.
4. Execute the supplied shell script "install" to create the TIP/ix system.
5. The install script prompts you for the directory where you want this TIP/ix system installed (the default is /usr/tipix). The release files are uncompressed and copied to the directory you specify. Make sure that the directory name is kept as small as possible, and that it is easy to remember.
6. This step should take approximately five to ten minutes.
7. You may also prompted to supply the serial number of the TIP/ix system and the "activation key" that is supplied by Inglenet Business Solutions with your copy of TIP/ix. Please enter this information very carefully.
8. Place the serial number(s) and activation key(s) in a safe place. This information is required every time you install TIP/ix!
9. Remove the temporary directory.
10. Logout — at this point it is not necessary to remain logged in as the "root" user. It is good practice to login as root only when it is strictly

necessary and use another (less omnipotent) user login at other times.

Example of TIP/ix installation dialogue

A box with *thin lines* is for an *initial installation*. A box with a *shadow* is for an *update*.

```
install: TIP/ix Release 2010/03/18 2.5 R0 - 0176 Installation
install: ================================================
install:
install: TIP/ix runs under its own user id and groupid for improved
install: system security. If you have not already done so you should
install: create this TIP/ix administrative user id before continuing.
install: The primary group for this administrative user id will be used as
install: the TIP/ix administrative group.
install:
install: Do you wish to continue with the installation? (<y>, n) > y
install:
install: Enter the path to TIP/ix system (</u/tipix>, q to quit) >
/u/tipix
install:
```

```
install: The TIP$SEC security values have been reviewed and
install: adjusted to more appropriate values. You may choose
install: to have these values installed or preserve your
install: current settings.
install:
install: Do you wish to install the new security values (<n>, y) > n
install:
```

```
install: This install script will copy in every component of TIP/ix.
install: Such components include the example source, development
install: libraries and conversion utilities which are part of the Heritage
install: Support Package.
install:
install: Do you wish to make any changes to the install list? (<n>, y) > y
install: Install the TIP/ix documentation files? (<y>, n) > y
install: Install the development environment? (<y>, n) > y
install: Install the database interface? (<y>, n) > y
install: Install the Heritage Support Package? (<y>, n) > y
install:
```

```
Install: Installing TIP/ix system /u/tipix from /tmp/tipix
```

```
Install: Updating TIP/ix system /u/tipix from /tmp/tipix
```

```
install:
install: The following components will be installed:
install: TIP/ix Base System
install: TIP/ix Documentation
install: TIP/ix Development Environment
install: TIP/dbi Database Interface
install: TQL/ix TIP Query Language
install: HSP/xx Heritage Support Package
install:
install: Do you wish to continue with the installation? (<y>, n) > y
install: Creating /u/tipix directory
```

```
TIP/ix System installation and initialization
TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business Solutions
```

```
System files in directory '/u/tipix/' Host node uw7test

Please enter the TIP/ix serial number
 or 0 if this is a Demo copy> 20300483
TIP/ix serial #20300483

For product TIP/ix serial number 20300483
Please enter the activation key> 9GGRLHQN

Enter your company name> Inglenet
Enter this system name> uw7test

The TIP/ix system now runs under its own user id and groupid,
rather than running as root. This makes for a more secure
system. Consult your release notes for further details.

You will need to create a new user id and groupid for this
TIP/ix system to run under, if you have not done so already.

What is the name of this TIP/ix system's administrative user id? > tipixusr
The TIP/ix administrative groupid will be "tipixgrp", okay? (<Yes>/no) > y

TIP/ix now uses the mmap(2) system call to map its shared
memory onto a file (rather than the system swap device).
This allows TIP/ix to use memory more effectively, and
eliminates the need for shared memory segments (shmop).
Consult your release notes for details about this important
new feature.

The default pathname for the mmap file is "/u/tipix/tipmmap".

Enter the pathname for the mmap file > [Return]
Using default path "/u/tipix/tipmmap".

For TIP/ix to operate, the file /u/tipix/tipmmap
needs to be created and sized to 778240 bytes. Is it all
right if I create it now?(<Yes>/no) > y

Writing mmap file /u/tipix/tipmmap...done.
Inglenet Business Solutions Registered for the following products on uw7test
  Concurrent Serial
Product Users Expiry Number
TIP/ix 10 2012/10 20300483
 TIP/ix administrative uid is tipixusr(400), gid is tipixgrp(103)
 Currency symbol character is -C $
 Decimal point character is -D .
 Activation Area shared memory -L 600K
 Global System shared memory -M 148K
 Global Data Area (GDA) -G 2K
 Memory Type (-TS=shm -TM=mmap) -TM (shared memory file "/u/tipix/tipmmap")
TIP/ix system control file built

TIP/ix System control files initialized
```

```
 TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business
 Solutions
 TIP/ix System Control, TIPROOT = /u/tipix/
 TIP/ix configuration /u/tipix/conf/tipix.conf missing parameters
 Please add PARAM LOCAP=<TIP/ix LOCAP name>

 TIP/ix System is not currently running
 install: Updating system; run tipinstall -i later if needed

 TIP/ix System installation and initialization
 TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business
 Solutions
 System files in directory '/u/tipix/' Host node uw7test
 Inglenet Business Solutions Registered for the following products on
 uw7test
```

```
  Concurrent Serial
Product Users Expiry Number
TIP/ix 10 2010/12 20300483
 TIP/ix administrative uid is tipixusr(400), gid is tipixgrp(103)
 Currency symbol character is -C $
 Decimal point character is -D .
 Activation Area shared memory -L 600K
 Global System shared memory -M 148K
 Global Data Area (GDA) -G 2K
 Memory Type (-TS=shm -TM=mmap) -TM (shared memory file
"/u/tipix/tipmmap")
```

```
install:
install: Installing base TIP/ix
install: Uncompressing TIP/ix modules for installation.
install: processing bin
install: processing conf
install: processing help
install: processing scripts
install: processing tipfiles
install: processing tipfe
install: Merging TIP/ix system files over to /u/tipix
install: processing tipmcs
install: processing tipsys
install: processing tipsec
install: processing tipmsg
install: Constructing basic security information
install: Copying bin
install: Copying help
install: Copying scripts
install: Copying TIP/fe binaries
install: Installing TIP/fe terminfo entry
install: Setting executable permissions
install: Fixing permissions on TIP/ix system files
TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business
Solutions
```

```
install: Creating default tipix.conf file
```

```
install: Removing obsolete items
install:
install: Installing TIP/ix documentation files
install: Uncompressing TIP/ix documentation for installation.
install: Copying TIP/ix documentation files
install:
install: Installing TIP/ix development environment
install: Uncompressing TIP/ix development modules for installation.
install: processing include
install: processing lib
install: processing src/tip
install: Copying COBOL copybooks and C headers
install: Copying libraries
install: Copying sample source code
install:
install: Installing TIP/dbi
install: Uncompressing TIP/dbi modules
install: Copying executables
install: Copying include files
install: Copying libraries
install: Copying examples
install:
install: Installing TQL/ix
install: Creating directory structure
install: Uncompressing TQL/ix modules
install: Copying executables
install: Copying examples
```

```
install: Copying program template
```

```
install: Copying record template

install: Setting TQL permissions
install: Removing obsolete items
install:
install: Installing HSP/xx
install: Uncompressing HSP modules
install: Clearing old executables
install: Copying executables
install: Copying configuration files
install: Copying examples
install: Copying include files
install: Copying scripts

install: Removing old TIP/fe binaries

install: Removing obsolete items
install:
install: TIP/ix installation temporary files are in /tmp/tipix
install: Do you want the TIP/ix installation temporary files removed?
(<n>, y) >
install: TIP/ix installation files remain in /tmp/tipix
install:
install: Remember to add TIPROOT=/u/tipix to your environment
install: Remember to add '/u/tipix/bin' to your PATH
install:
install: Installation is complete
```

Initial System Setup

Before using the installed TIP/ix system, at least two environment variables must be correctly set:

- TIPROOT the path where the TIP/ix system files are located
- TERM the terminal type

Set the environment variable TIPROOT to the directory in which you installed the software (for example: /usr/tipix).

Add the definition of the TIPROOT environment variable to your login startup file (".login"), or ".cshrc" or "/etc/profile". Your system administrator can help you to define environment variables.

The following commands are used if you are using the C-shell:

```
setenv TIPROOT "/usr/tipix"
```

If you are using the Bourne or Korn shell, the syntax is:

```
TIPROOT="/usr/tipix"
export TIPROOT
```

You should also update each users' definition of the PATH environment variable to include **$TIPROOT/bin**. Each TIP/ix user may also set the environment variable **TIPPATH** to define where TIP/ix executables are located.

The TERM environment variable must be set to define the type of terminal that you are using to interact with TIP/ix. For now, if you intend to

use the supplied TIP/fe program, set the TERM variable to "tipfe" otherwise, "vt100" or "tipvt" is suggested.

If you require special EBCDIC to ASCII character translation to support, for example, national character sets, see the **ebcasc** utility program for instructions on how to specify such a special translation.

tipinstall - System Administration

The install script automatically executes the **tipinstall** program using default values for various TIP/ix system parameters. The default values should suffice for most sites. This section describes how to run the **tipinstall** program to alter the TIP/ix system parameters. The **tipinstall** program can be run at any time (certain operations require TIP/ix to be shutdown first.)

If you are installing a full usage version of TIP/ix, Inglenet will supply you with an activation key. When asked for the serial number, enter the serial number sent with the software.

You will then be asked for an activation key. The activation key is an eight character alphanumeric value. The activation key validates the software license. If you upgrade your TIP/ix license, you will be supplied with a new activation key. Store the activation key and serial number in a secure place!

The **tipinstall** program accepts the following command line options:

Option	Description
-C$	MCS currency symbol. Default: "**$**".
-D*d*	The default is determined by TIP/ix by examining the system environment variable **TZ** (time zone): . North American default is a period (ASCII X'2E'). , European default is a comma (ASCII X'2C').
-G*n*	"*n*" is the size (in bytes) of the TIP/ix Global Data Area (GDA). Minimum size is 256 bytes. All sized rounded off to nearest 256 bytes. **Note:** The new GDA size is not automatically reflected in tipix.conf. This causes wrong GDAsize to be reported as the GDAsize value in tipix.conf takes precedence during settings report. A quick fix for this is to manually change the GDAsize to reflect the new size. However, if you do not have any entry for GDAsize in the tipix.conf, tipinstall reports the correct size.

Option	Description
-i	Initial license information. This option is used to apply a new activation key (serial number and/or number of users) to the TIP/ix system. (Use **-n** for updating the license key.)
-l*c*	Language code. The default is American English. Possible codes are: A American English B Canadian English C Canadian French D Dutch E UK English F French G German H Swiss-German I Italian K Afrikaans N Norwegian P Portuguese R Swiss-French S Spanish W Swedish X Danish Y Finnish
-L*n*	Size of the Activation Area shared memory. This area is a block of shared memory used to store the activation records (PIB, CDA, MCS, WORK, IMA, OMA, AFT etc.) for all executing TIP/ix programs. The default, computed by TIP/ix, should suffice for most systems. Approximately 24K bytes are required for each interactive user and background process. The value 'n' can be a number optionally followed by K (kilobytes) or M (megabytes). For example, 1200M means 1200 megabytes. See ***UNIX Kernel Parameters*** section in this book. If n = 0 then this tells TIP/ix to decide on the appropriate size of the area itself. Although to do this it checks to see what value was given to the GDASIZE in the tipix.conf file.

Option	Description
-M*n*	Global system shared memory. The default, computed by TIP/ix, should suffice for most systems. The value '*n*' is a number optionally followed by K (kilobytes) or M (megabytes). For example, 150K means 150 thousand bytes. If the value specified is too small, you could run out of memory and get the PIB-FULL error. However, if it is too high, TIP/ix will consume more memory than it needs and this may degrade overall system throughput. To determine how much -M memory is used, let TIP/ix run for a while then execute **status s**. See ***UNIX Kernel Parameters*** section in this book.
-n	Establish new license key. You are prompted for the product name, serial number, and activation key. For TIP/ix the product name is 'TIP/ix'.
-T*t*	Memory type, as follows: S To use UNIX shared memory. M to use a memory mapped file. On systems which support memory mapping, this is the preferred choice.
-u	Display some recommendations for minimum values for the related UNIX kernel parameters based on the single TIP/ix system. These recommendations do not take into consideration any other products which may be using the UNIX system. See ***UNIX Kernel Parameters*** section in this book.

The following options have been replaced by parameters in tipix.conf:

-Q Replaced by FCSQUEUES

-U Replaced by USERQUEUES

The TIPINSTALL program can be executed without command line options to receive a report on the current settings:

Note: All memory allocation rounded off to the nearest 256 bytes.

```
$ tipinstall
TIP/ix System installation and initialization
TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business
Solutions
 System files in directory '/u/tipix/' Host node uw7test
Inglenet Business Solutions Registered for the following products on
uw7test
  Concurrent Serial
Product Users Expiry Number
TIP/ix 10 2012/10 20300483
```

```
TIP/ix administrative uid is tipixusr(400), gid is tipixgrp(103)
Currency symbol character is -C $
Decimal point character is -D .
Activation Area shared memory -L 600K
Global System shared memory -M 148K
Global Data Area (GDA) -G 2K
Memory Type (-TS=shm -TM=mmap) -TM (shared memory file
"/u/tipix/tipmmap")
$
```

Copying Configuration Sets

TIP/ix sites often define *sets* to group related configuration items by name. For example, you could define a "payroll" set for the payroll application system.

The TIP$SYS file contains TIP/ix configuration records including information about: user ids, programs, files, queues, LOCAPs, terminals, and groups. Each record in the TIP$SYS file has a *set* field to identify related configuration entries. The tipinstall program can be used to copy all records for a set from one TIP$SYS file to another.

There are two steps to copy configuration records between TIP/ix systems.

- Use tipinstall to create a temporary file of all records for a given set.
- Use tipinstall to merge these records into a TIP$SYS file for another TIP/ix system.

Syntax:

```
tipinstall [-S setname] [-r dir] -g input_tipsys
tipinstall [-S setname] [-r dir] -s input_tipsec
```

Parameters:

setname
: The name of the SET to read. All definitions associated with this set name are read.

dir
: The name of the directory where the temporary tipsys.xxx file is to be created. This directory must exist; tipinstall will not create the directory.

input_tipsys
: The input tip system file.

input_tipsec
: The input tip security file.

Example 1:

Copying all System Set information to temporary file:

```
tipinstall -S ARC -r temp -g $TIPROOT/tipfiles/tipsys
```

- *This command gathers all definitions from the input TIPSYS file: $TIPROOT/tipfiles/tipsys (specified via -g) for the set ARC (via -S),*

and writes them to the indexed file with components named "tipsys.dat" and "tipsys.idx" in the directory temp (-r).

Example 2:

Copying all Security Set information to temporary file:

```
tipinstall -S ARC -r temp -s $TIPROOT/tipfiles/tipsec
```

- *This command gathers all definitions from the input TIPSEC file: $TIPROOT/tipfiles/tipsec (specified via -s) for the set ARC (via -S), and writes them to the indexed file with components named "tipsec.dat" and "tipsec.idx" in the directory temp (-r).*

Example 3:

Merging System definitions:

```
tipinstall -g /temp/tipsys
```

- *This command reads the definitions from the file specified by the (-g) option and writes them into the current $TIPROOT/tipfiles/tipsys file.*

Example 4:

Merging Security entries:

```
tipinstall -s /temp/tipsec
```

- *This command reads the definitions from the file specified by the (-s) option and writes them into the current $TIPROOT/tipfiles/tipsec file.*

Contents of the TIPROOT Directory

After installing the software, the directory defined by the environment variable TIPROOT (/usr/tipix for example) contains the following subdirectories:

Directory	Description
bin	holds the executable binary programs. Add this directory to your PATH environment variable.
conf	holds all configuration files such as `tipix.conf session.conf, share.conf` and `tipkeys.sys`
cpy	`tipcblxml` write COPY books here for Web Service programs
docs	Currently this is not used. Documentation is provided as PDF files which can be downloaded after logging into the Inglenet website.
include	holds the TIP/ix COBOL copybooks and C language header files.

Directory	Description
help	holds the TIP/ix "help" source files. These files are in the same format as the TIP/30 help files.
lib	holds the development libraries released as part of TIP/ix
log	holds system log files created during system execution
scripts	holds sample and utility scripts
src/arm	holds Heritage Support Package related samples
src/tip	holds some sample TIP/ix COBOL programs and make files.
src/tql	holds sample TQL programs and a TIP/ix script to automatically load the TQL system
tipfe	holds copies of TIP/fe and related support files. Currently this is not used.
tipfiles	holds the TIP/ix system files and a sample data file for the sample programs.
tipsoa	Hold XML-COBOL data mapping files for the SOAP/XML interface
tmpwrk	holds inbound DTP session log files, and is also the location of log files for programs that run in the background.
tql	holds the TIP/ix TQL (Query Language) system files.

Notes:

- You should not put your own files under $TIPROOT or update our scripts as we write over them with every release.
- Any security entries for transactions with the same name as a TIP transaction will be overlaid with the TIP transaction setting. For example, if you have a transaction named MENU, then after installation you will have to change the SMSEC entry for MENU to refer to program definition for your own menu program.

The old files can be removed immediately if you choose not to create the compatibility links. If you have copied any of your own files to one of the directories in the above table you may want to postpone the clean up until a later date, then move these local files by hand.

TIP/ix System Files

The following files are all listed relative to $TIPROOT:

File	Description
log/CRASH	is created if the TIP/ix system crashes for any reason. This information may be helpful to Inglenet customer support.
log/ERRORS	may contain messages for serious errors which occurred in a system process which did not otherwise have a log file.
log/HISTORY	tipctl keeps a log of TIP/ix startups and shutdowns, including time, user id and action.
log/PRINTER	Collects output from transaction programs DISPLAY UPON PRINTER
log/tipfcs0	is created when the TIP/ix system is started. This file will hold any error messages from the TIP/ix File Control System. You may want to list it periodically to see if there are any error messages.
log/tipfcs1	The TIP/ix file control system operates as several processes (up to 64) to balance the I/O and improve system throughput. Each FCS process has its own log file called log/tipfcs1 through log/tipfcs64.
log/tipmon	is created when the TIP/ix system is started. This file will hold any messages from the TIP/ix monitor process. They are mostly informational. If the TIP/ix system seems to be malfunctioning then check the contents of these **log** files for possible error messages.
log/tippcstm	is created when the TIP/ix system is started. It holds information related to the management of re-usable transaction programs (usually IMS) and database interface server processes.
log/tipque	is created when the TIP/ix system is started. It holds information related to the operation of the TIPQUEUE function.

File	Description
log/tipdtp	is created when the TIP/ix system is started. It holds information related to distributed transaction-processing activity.
gda.mem	is created if the TIP/ix system crashes. It contains a copy of the GDA memory area. For use by Inglenet only.
global.mem	is created if the TIP/ix system crashes. It contains a copy of the global memory area. For use by Inglenet only.
local.mem	is created if the TIP/ix system crashes. It contains a copy of the local memory area. For use by Inglenet only.
tipboot.sys	is created by the **tipinstall** program. This file contains system information and is used to keep track of whether TIP/ix is currently running and which system resources are being used.
tipmmap	is created by the **tipinstall** program. This is used with UNIX memory mapping to establish shared memory.
tipfiles/tipix.abt	is a log of all aborted distributed transactions by Global Transaction ID. TIP/ix uses it during startup recovery.

TIP/ix Data Files

If the TIP/ix root directory is "/usr/tipix" then the directory "/usr/tipix/tipfiles" contains the following files:

File Name	Description
tipix.jrn0	Journal file(s). Specified with one or two JRNFILE= parameter(s) in tipix.conf.
tipix.qbl0	"Quick before look" file. Specified with up to six QBLFILE= parameters in tipix.conf.
tipixmsg.a	Canned messages for TFD in American English
tipixmsg.f	Canned messages for TFD in French
tipiximage	Compiled message file for TFD

File Name	Description
tipmcs.*	File holding MCS formats
tipmsg.*	Canned message file for TIPMSG.
tipque	Storage file for TIPQUEUE messages.
tipsec.*	TIP/ix security information used to control access to files, programs, and queues
tipsys.*	TIP/ix configuration information for user ids, files, programs, queues, group sets, terminals, printers, and LOCAPs
tiptrm.*	TIP/ix index to terminal configuration records in tipsys.
*.ps	Postscript programs.

Notes:

- Specifying two JRNFILE= parameters (each with a unique filename), enables journal file swapping. For manual swapping, see the jrnswap utility. For automatic swapping, see the JRNSIZE= parameter. For an overview of swapping, see the jrnswap utility.
- If the JRNFILE= parameter does not specify a path, the journal file will be created in $TIPROOT.
- You can *improve system performance*, by specifying *up to six qbl files*, especially if the qbl files are *on different devices*. Each qbl file must have a unique name.

Relocating TIP/ix System to a new computer

Over time, you may find a need to acquire new hardware and that may require you to move your current TIP/ix system to a new Unix/Linux system. There are no hard rules since each system is unique. Here are some guidelines for doing this.

Create all Unix/Linux group names that you use and assign them the same group id number as is used on the current system. See man page for groupadd for details. (See /etc/group for what these are.)

Create all Unix/Linux userids on the new system with the exact same userid number as was used on the current system. (See /etc/passwd for what these are.) Refer to the 'man' page for useradd for details on how to do this.

Setup your new system with a similar (if not identical) file structure. This will avoid many problems of file paths and file names that may be hard coded in scripts or control files.

Install other needed software such as the COBOL compiler and maybe the database (such as Oracle) if this is being used.

Copy over all user home directory contents (complete copy). You need to be 'root' to do the file copies so that the files on the new system get set to the correct permissions. You could copy using any number of methods. A simple method is to use tar to collect many directories and files into a compressed tar file, then ftp that to the new system and then use tar to extract the contents into the right location.

Copy over all data files, source code, control files etc to the new system. Remember to copy any CVS repository and repair any CVS control files if you change location of the CVS repository.

For TIP/ix, this would be a good time to upgrade to the latest version of TIP/Ix. The best way to do this is to copy of the entire contents of the current TIP/ix system (starting at the $TIPROOT directory) to the new system and then install the latest TIP/ix software over top of that. (The most important point is to copy the files from the $TIPROOT/tipfiles directory to the new $TIPROOT/tipfiles on the new system. The TIP/ix control files that define transactions, users and data files are in this directory.) It must be done in this order so that the control files from the old system are copied in first. Then installing the new TIP/ix software retains the control information but does update the software modules and makes any needed additions to the control files.

Ideally you should use the same Unix userid & groupid numeric values on the new system but if that is not possible for any reason you can repair the TIP/ix system by running: `tipinstall -Z`

The –Z option to tipinstall will ask you for the new Unix user to be the TIP/ix administrator. Answer the prompts. Then tipinstall program will repair the file and directory permissions for the TIP/ix system. You will have to manually repair the permissions for your own data files.

TIP/fe - TIP/ix Front End

What is TIP/fe

TIP/fe is a stand-alone terminal emulator *and* a front-end interface for TIP/ix. When running TIP/ix, you can use TIP/fe in either vt mode or in smart mode:

- In *vt* mode TIP/fe operates as a VT220-like terminal emulator.
- In *smart* mode, TIP/fe operates as an intelligent display server.

TIP/ix delegates all MCS related duties to TIP/fe. TIP/fe does all the MCS processing on your desktop, freeing up the UNIX CPU to perform other tasks, thus improving overall system performance.

When running TIP/ix through TIP/fe, you select the TIP/fe mode by setting the UNIX TERM environment variable:

- Set TERM to *tipfe*, for smart mode.
- Set TERM to *tipvt*, for vt mode.

TIP/fe can compress *smart* mode packets that are sent over the network. Compression is especially important if you have slow data lines, or there is a lot of network traffic.

Another important advantage is that TIP/fe is “TIP/ix aware”. It can provide a keyboard layout which is largely compatible with UNISCOPE terminal emulators that are available for the Unisys 2200 such as Attachmate’s INFOCONNECT products.

TIP/ws (Work Station)

TIP/ws is a newer replacement for TIP/fe. It provides all of the same functionality as TIP/fe plus some advanced features. TIP/ws is better integrated into the MS Windows philosophy.

One difference between TIP/fe and TIP/ws is that TIP/fe uses a ‘multi-document’ (or multi-screen) design while TIP/ws uses a ‘single-document’ per session.

TIP/ws is documented in detail in a separate document.

TIP/fe for Windows

TIP/fe for Windows can run as a VT220 terminal emulator or as a smart front-end to TIP/ix.

Pointer to Installation Instructions

The ***TIP/ix Release Notes*** contain the installation instructions for TIP/fe.

Read the **README.DOC** file on the installation disk ***before*** attempting any installation. This file contains any last-minute changes not in the release notes.

Starting TIP/fe

To start TIP/fe, double click on the TIP/fe icon. If TIP/fe cannot find the appropriate communications drivers, a dialog box with an appropriate message will be displayed on your screen.

If no hosts have been defined to TIP/fe, you will be prompted with the 'TIP/fe Configuration' dialog box to define the host(s) you will be connecting to.

On-line Help

The TIP/fe program has on-line help to explain its operation and configuration.

Session Control

TIP/fe can have multiple sessions open concurrently with each session functioning in terminal emulation mode or in MCS server mode for TIP/ix. An initial session is opened when TIP/fe is loaded and subsequent sessions can be opened or closed as desired.

To open a session, use the ALT-O key or ALT-F#, where # is a number between 1 and 8 representing a function key. If that session is not already opened, you will prompted to enter a hostname and then to log in. If that session is already open, you will move to that session.

To close a session, use the UNIX standard logout or exit command so the host system will handle a clean disconnect. If the session can not be closed this way, you can use the ALT-K key to 'kill' the session, but only as a last resort.

To display a list of the open sessions, use the ALT-L key to show a summary with the status of each.

To switch to the next open session in sequence, use the ALT-N key. These session control keys are summarized under 'Hot Keys' later in this section.

Custom Colors

When at least one session has been opened, you may use the ALT-C hot key to open a dialog box to change the specification of the colors that

TIP/fe uses for various UNIX modes or MCS screen attributes. The following sample screen is, of course, black and white. The colors you will see when you display this screen depends on the type of monitor you are using.

TIP/fe allows you to select the colors to be used for each purpose. When TIP/fe terminates, the file *tipfe.cfg* is created in the directory specified by the MS-DOS environment variable **TIPFE**. If no such environment variable is found, the configuration file is created in the root directory of the C: drive.

Mouse Handling

When TIP/fe is connected to a TIP/ix system and the terminal type is defined as TIPFE (that is, TERM=TIPFE), the mouse can be used to perform additional actions. The left mouse button can be double-clicked to generate a transmit key operation. While the cursor is in a TIP/ix screen field, you can double-click the right mouse to request field-level help information. The help information will only be displayed if such information is defined for the field in the screen format.

Hot Keys

While TIP/fe is running, the character sequence ALT-H provides the following help summary for the hot keys that TIP/fe recognizes:

Hot Key	Description
ALT-F#	Pressing ALT and a function key from F1 through F8 inclusive causes TIP/fe to go to the corresponding session (1 through 8). If that session is currently closed, TIP/fe opens a dialog box to prompt the user for the host name for that session and then switches to that session's screen.
ALT-A	This hot key is used to configure printer destinations. TIP/fe grabs UNIX output sent to AUX devices and reroutes the output to the destination as specified on this screen. This can be a file or printer attached to the computer.
ALT-B	This hot key is used to end record mode or replay mode.
ALT-C	This hot key is used to customize colors that TIP/fe uses.
ALT-F	This hot key is used to Flip to the last active session.
ALT-H	This hot key is used to display a help screen for TIP/fe hot keys.

Hot Key	Description
ALT-K	This hot key is used to 'kill' (terminate) a session with a UNIX host system*. It should be used only as a last resort*. It is preferable to terminate the session with a standard logout or exit command so that the host system observes a clean disconnect.
ALT-L	This hot key causes TIP/fe to pop up the following dialog box, showing a summary of the 8 possible sessions and the status of each session. From this dialog box you may select a session to go to by pressing the corresponding session number. Press any other key to close the dialog box and return to the previous session.
ALT-M	This hot key causes TIP/fe to display a dialog screen that allows you to specify the preferred character set (American, French etc.). The default is the American character set. The following is an example of the dialog screen:
ALT-N	This hot key causes TIP/fe to move to the next open session. If windowing has been set on, this hot key proceeds to the next session that has not been minimized (set to an icon).
ALT-O	This hot key is used to open a new session with a UNIX host system. The next available session number in sequence from the current session number is selected and the user is prompted for the hostname.
ALT-P	This hot key inserts (pastes) text that has been previously marked. The text is pasted as if it was typed from the keyboard at the cursor location. A carriage return is supplied at the end of each line except for the last (or only) line of keystrokes. Lines of text can be marked by holding down the left mouse button while traversing the lines to be marked.
ALT-Q	This hot key attempts to close all active sessions and terminate TIP/fe. This should be considered a crude way to shutdown all active sessions rather than closing each session individually and, when no active sessions remain, having TIP/fe terminate normally.

Hot Key	Description
ALT-R	This hot key turns on record mode. TIP/fe prompts you to enter the name of a DOS file where the session record is to be written. From this point, TIP/fe records in the specified file all keystrokes that you enter and information about the screen formats used and responses. ALT-B cancels record mode.
ALT-S	This hot key is used to take a snapshot of the current screen.
ALT-T	This hot key is used during replay mode to cause TIP/fe to show you the screen layout that is expected. Press any key to continue with replay mode.
ALT-W	This hot key is used to toggle between windowed mode and full screen mode
ALT-Y	This hot key turns on keyboard replay mode. TIP/fe replays all the keys recorded with a prior ALT-R function and will wait for the screen layout to exactly match the layout at the time the keystroke was pressed. During screen layout matching, TIP/fe recognizes date and time fields and ignores them. When in replay mode, any discrepancy in the layout of the screen and the keys causes replay mode to stop. ALT-B cancels replay mode.
ALT-Z	This hot key spawns a copy of the MS-DOS command interpreter (a DOS shell) if enough memory is available. Use the DOS ‘exit’ command to return to TIP/fe.

For TIP/ws running TIP/ix in smart mode.

Hot Key	Description
CTRL-V	Inserts a space.
CTRL-Y	Deletes one character at a time.
CTRL-W	Toggles between OVR and INS

For TIP/ws running TIP/ix in TTY mode.

Hot Key	Description
CTRL-Y	Deletes one character at a time.

For TIP/fe running TIP/ix in smart mode.

Hot Key	Description
CTRL-Y	Deletes one character at a time.
CTRL-W	Toggles between OVR and INS

For TIP/fe running TIP/ix in TTY mode.

Hot Key	Description
CTRL-V	Inserts a space.
CTRL-Y	Deletes one character at a time.

Internationalization for DOS

In some countries, a mapping table is needed to map local characters onto characters recognized by the UNIX host. TIP/fe for DOS cannot automatically sense the country it is running in, and the mapping table should be set through the ALT-M hot key command.

TIP/fe supports twelve standard languages sets, where up to 12 local characters are mapped onto characters supported by the UNIX host.

Customized character mapping is supported through two additional conversion tables, so you can edit character mappings as needed.

In general, *you should* set the MS-DOS code page correctly for the language you are using. You *can* use MS-DOS code page 437 and 850 for *most* languages, but some languages *require* local characters. For example:

- Norwegian needs characters that code page 437 lacks, so use 865.
- Portuguese needs characters that 437 lacks, so use 860.

Supported Languages

TIP/fe supports twelve standard language sets, where up to 12 characters recognized by the UNIX host are replaced with characters needed for a specific language. The default mappings are:

Character Set		23	40	5B	5C	5D	5E	5F	60	7B	7C	7D	7E
English (American)	enu	#	@	[	\	]	^	_	`	{	|	}	~
English (International)	eng	£	@	[	\	]	^	_	`	{	|	}	~
Canadian French	frc	#		â	ç	ê	î	_	ô	é	ù	è	û
Danish	dan	#	@	Æ		Å	^	_	`		ƒ	¼	'
Finnish	fin	#	@	Ä	Ö	Å	Ü	_	é	ä	ö	å	ü
French	fra	£		°	ç		^	_		é	ù	è	
German	deu	#		Ä	Ö	Ü	^	_	`	ä	ö	ü	ß
Italian	ita	£		°	ç	é	^	_	ù		ò	è	ì
Norwegian	nor	#	@	Æ		Å	^	_	`	æ	ø	å	~
Portuguese	ptg	#	@	Ã	Ç	Õ	^	_	`	ã	ç	õ	~
Spanish (Modern)	esn	£		¡	Ñ	¿	^	_	`	°	ñ	ç	~
Swedish	sve	#	É	Ä	Ö	Å	Ü	_	é	ä	ö	å	ü
Swiss	sws	ù		é	ç	ê	î	è	ô	ä	ö	ü	û

The Swiss character set includes a mapping into character '_', which could cause difficulties in programs where data fields are initially filled with this character.

Language Awareness

In *smart mode,* TIP/fe (in addition to performing character mapping) becomes language aware. This is needed as, in smart mode, TIP/ix delegates all MCS related duties to TIP/fe.

Being "language aware" for TIP/fe means that data field definitions, such as Alphabetic, Alphanumeric or Uppercase field, become language sensitive. Thus, alphabetic and uppercase field definitions are expanded

appropriately for each particular language, including alphabetic characters listed in the above table.

Uppercase conversions are performed in accordance with common conventions for particular country (for example, when French language is selected, accents are dropped in monocasing, that is, lower case characters ‘é’ and ‘ç’ would be translated into uppercase ‘E’ and ‘C’, respectively. In Swedish and Portuguese those two lower case characters would be appropriately mapped into accented uppercase characters).

Locals other than the ones listed in the above table are not recognized. Thus, for example for a language selection ‘French’, the accented lowercase character ‘é’ would be treated as an alphabetic character, but the same upper case character É would not be recognized, since it cannot be found in the mapping table for French (character ‘é’ is by default mapped into ‘E’).

In *TTY mode*, TIP/fe is not language aware, and does not follow common conventions for numeric fields, based on a country selection.

Terminal Interface

Using the Terminal Interface

When you are using a transaction program that uses the terminal interface (that is, MCS screen formats), TIP/ix simulates a traditional UTS terminal. Data is passed to the transaction program when you press either a function key or **transmit**. Transmit is indicated by pressing the **PGDN** key.

You may also specify that you want the transmit key equated to the RETURN or ENTER key, by using the SMUSER transaction.

An exception is when the application has requested field level input. In this case when the end user changes the contents of a data field and moves the cursor out of that field and the application requested field level input for the field in question, all data on the screen up to that point is returned to the application and a status of MCS-F-FIELD is indicated in the application program.

If the terminal has a full keyboard and the "curses" terminal definition defines all of the keys, TIP/ix will correctly handle all function keys and cursor movement keys. To support many different terminals TIP/ix also accepts various CTRL key combinations (see table on following pages).

In the Supported Keyboard Sequences table, you must press keys that are hyphenated *simultaneously*. For example, the notation CTRL-F 1 means hold the CTRL key while pressing the F key, then release both CTRL and the F key and press 1.

It is important to realize that keyboard handling is primarily a function of the host UNIX system. The situation is further complicated by the presence of "terminal emulators" that often have their own interpretation of the key sequences desired when a particular key is pressed. For additional information about this topic, see *artie Terminfo File Editor*.

Supported Keyboard Sequences

The following table summarizes the keyboard sequences supported by TIP/ix. TIP/ix recognizes a primary key sequence for most functions and permits an alternate definition for terminals that lack some common keys. Also included in this table are additional key definitions that are recognized by TIP/fe (a UNISCOPE terminal emulator for MS-DOS that is included with TIP/ix). TIP/fe keyboard mapping has these additional values to accommodate former users of other UNISCOPE terminal emulators such as those supplied by Computer Logics Corporation (PEP):

Function	TIP/ix Primary	TIP/ix Alternate	TIP/fe Alternate
Backspace	backspace	CTRL-H	
Previous Field	Back Tab	CTRL-R	Shift-TAB
Next Field	Tab	CTRL-I	
Beginning of Form	Home Home	CTRL-B CTRL-F Home	
Clear Line/Field	CTRL-E		ALT-9
Delete Character	del	CTRL-Y	
Delete Line	CTRL-K		
Down Arrow	CTRL-J		
End of field	end		
Start of field	home		
Insert/Overtype (toggle)	ins	CTRL-W	
Insert Line	CTRL-L		
Next Page	page down	CTRL-N	

Function	TIP/ix Primary	TIP/ix Alternate	TIP/fe Alternate
Previous Page	page up	CTRL-P	
Restore Field Contents	CTRL-F backspace	page up	
Return	Enter	CTRL-M	
F1	F1 CTRL-F 1		F1
F2	F2 CTRL-F 2		F2
F3	F3 CTRL-F 3		F3
F3	F3 CTRL-F 3		F3
F3	F3 CTRL-F 3		F3
F3	F3 CTRL-F 3		F3
F4	F4 CTRL-F 4		F4
F5	F5 CTRL-F 5		F5
F6	F6 CTRL-F 6		F6
F7	F7 CTRL-F 7		F7
F8	F8 CTRL-F 8		F8
F9	F9 CTRL-F 9		F9
F10	F10 CTRL-F 0		F10
F11	CTRL-F q		Shift-F1
F12	CTRL-F w		Shift-F2
F13	CTRL-F e		Shift-F3
F14	CTRL-F r		Shift-F4
F15	CTRL-F t		Shift-F5
F16	CTRL-F y		Shift-F6
F17	CTRL-F u		Shift-F7
F18	CTRL-F i		Shift-F8
F19	CTRL-F o		Shift-F9
F20	CTRL-F p		Shift-F10

Function	TIP/ix Primary	TIP/ix Alternate	TIP/fe Alternate
F21	CTRL-F a		CTRL-F1
F22	CTRL-F s		CTRL-F2
MSG WAIT	Esc	CTRL-[	Alt-1
Transmit	PageDown CTRL-] CTRL-F enter		Scroll Lock
Start of Entry	CTRL-O		Alt-2
Erase Screen	CTRL-F z		Alt-=
Duplicate Line	CTRL-D		Alt-ins
Activate Menu Bar	CTRL-\		
Display Help	CTRL-F h	CTRL-F ? CTRL-F /	
Tab Set	CTRL-T		Alt-tab
Screen Refresh	CTRL-f P	CTRL-f.	
End of Form	end end CTRL-F end	CTRL-N	
Beginning of Field	CTRL-F b		
End of Field	CTRL-F n		
Abort Program	CTRL-A	CTRL-C	
Print Screen			Print Screen
Pause Display			Pause
Display Control Page	CTRL-F c		Alt-3

Terminal Status Line

If you are using a terminal that displays at least 25 lines, the TIP/ix terminal interface status line is displayed on the bottom line. For many ASCII terminals with only 24 lines the status line cannot be displayed all of the time. The status line has the following format:

```
rr,cc mode ll type ESC ? hh:mm term TIP/ix ver
```

Where:

rr the row number of the cursor.

cc the column number of the cursor.

mode **INS** if you are in insert mode. Toggle insert mode and overwrite mode by using the **INSERT** key or press **CTRL**-**W**. If you are in overwrite mode and some MCS format windows have been pushed on the stack, this area displays the current window number; for example: "W02".

ll the length of the current field.

type the current field type: "Alpha", "Num", or "AlNum".

hh:mm
the current time of day.

term the terminal name.

ver is the TIP/ix version number.

TIP/ix Terminal Definition

When TIP/ix is installed, part of the installation procedure adds custom terminal definitions to the Unix system terminal definition database (terminfo database). These terminal definitions are:

- tipfe
- tipwfe
- tipvt

These definitions provide an enhanced version of the standard vt-100 terminal emulation. Many Unix implementations have a minimalist definition of a vt-100 terminal (for example, indicating that the vt-100 only has function keys 1 through 4). The standard definition is too restrictive for use with TIP/ix; the Tip terminal types essentially extend the vt-100 definition to allow all the function keys that TIP/ix can use (F1 through F22).

Some Unix programs (especially system administration programs) may have their own special requirements for terminal definition and may not work properly if the terminal type is set to anything they do not recognize or support. TIP/ix, however, will work correctly with any of the terminal types listed above.

These Tip terminal types are terminal definitions that allow the user to treat the keyboard in the same manner as the functionality provided by older PC-based terminal emulation products like PEP from Computer Logics. All of the terminal definitions listed above are identical; there are multiple names supplied for historical reasons (at one time, tipwfe has some unique mappings, but this is no longer the case).

The TIP Workstation terminal emulator (and the older TIP/fe product) both use the terminal name (in the TERM environment variable) to decide whether or not to operate in what is called "Smart Mode". In "Smart Mode" (TERM=tipfe or TERM=tipwfe), the terminal emulator running on the PC will perform many screen formatting functions locally on the PC rather than relying on processing at the host (Unix) system. This can result in considerable improvement in response time and enhanced functionality (for example, some TIP/ix screen calls may be displayed using familiar Windows controls rather than traditional character based output.

If the Unix system terminal database ever is corrupted or restored to a state that does not include the TIP/ix terminal definitions, those definitions can be recreated by following this procedure:

1. login as root
2. run the following command:

   tic $TIPROOT/include/tipfe.ti

The above command line assumes that the environment variable TIPROOT has been correctly set to the location where TIP/ix has been installed. This command runs the "terminfo compiler" (tic) that reads a terminal definition file (in tipfe.ti). Of course, the use of the "tic" program may have specific requirements depending on the particular Unix system used; consult the man pages for more information about tic.

For more information about how the terminal information database is constructed and used, please refer to the book: "termcap & terminfo" by John Strang, Linda Mui and Tim O'Reilly, published by O'Reilly & Associates (ISBN: 0-937175-22-6).

As part of the Unix system there may also be a utility called "untic" or "infocmd" that will extract the current definition of a specified terminal type into a file that can be edited and then submitted to the "tic" program. For information about these utilities consult the man pages for your Unix system. Not all Unix systems provide an easy means to extract information from the terminfo database; that is the reason the artie program is supplied as part of the general Unix utilities provided with the TIP/ix system.

TIP/ix uses the UNIX "curses terminfo" to interface with all terminals. This is controlled by the value of the **TERM** environment variable. The UNIX administrator should make sure that you are using the correct TERM value for your terminal. See the **artie** utility in the ***TIP/ix Utilities*** **manual.**

TIP/ix Shell Prompt

When you are in the TIP/ix shell, the prompt looks like the following:

```
TIP/ix?>
```

If TIP/ix was started in debug mode, the TIP/ix prompt looks like this:

```
Debug:TIP/ix?>
```

If you execute a UNIX transaction while in the TIP/ix shell, TIP/ix outputs the following message together with the prompt:

```
Enter the next UNIX command or Enter nothing to return to full screen
mode
TIP/ix?>
```

After running a UNIX command in the above fashion, you may encounter some difficulties with function key settings at this point. To reset the prompt, press **XMIT** or execute a TIP/ix transaction.

TIP/ix Data File Types

Indexed files

In batch COBOL programs define the file as **ORGANIZATION IS INDEXED**. In TIP/ix the file is defined as **ISAM** or **MIRAM**.

Each of the COBOL compilers have a different indexed (or ISAM) file format, so it is very important to know which COBOL compiler is being used for batch programs which may be accessing the data files. OS/3 customers may be used to the name MIRAM. In this discussion, an ISAM file in UNIX is identical in functionality to OS/3 MIRAM.

The various ISAM modules have different file naming conventions. The following table shows which files are created for an "ISAM" file named "myfile".

ISAM	Data	Index
Micro Focus	myfile	myfile.idx
C-ISAM	myfile.dat	myfile.idx
D-ISAM	myfile.dat	myfile.idx

Note: Micro Focus uses the C-ISAM file structure but it does not add the ".dat" extension to the data file. This is why TIP/ix creates a Unix link file that links the name "myfile" to "myfile.dat". This link file is needed so that both

Micro Focus COBOL batch programs and TIP/ix can access the ISAM file.

The COBOL-IT compiler uses VB-ISAM for the ISAM file structure is very different from that used by Micro Focus. TIP/ix recognized which ISAM file format is being used and handles each correctly.

Refer to the TIP/ix utilities manual for the ‘isreorg’ program which can be used to convert ISAM file formats.

ISAM files and Micro Focus vs D-ISAM

Warning: The latest version of TIP/ix uses a D-ISAM version 7 and the versions of Micro Focus 5 and later do not play well with D-ISAM. To avoid file corruption problems compile batch programs with Micro Focus COBOL but link them with the DISAM library.

DISAM has a utility called dcheck for verifying file format and dpack for rebuilding an ISAM data file.

If you are using a script then add compile options similar to the following to the cob command:

```
-L $TIPROOT/lib -m ixfile=cixfile +l disam
```

Inside a ‘makefile’ the syntax would be more like:

```
-L $(TIPROOT)/lib -m ixfile=cixfile +l disam
```

Sequential files

In batch COBOL programs define the file as **ORGANIZATION IS SEQUENTIAL**. In TIP/ix the file is defined as **SAM** or **SMIRAM**.

The file naming conventions used here are consistent. The file name is used as supplied and no extensions are added.

For Micro Focus COBOL a file of **n** byte records is just a series of **n** byte record slots. There are no special characters used to separate records. There is no way to mark records as deleted.

Relative files

In batch COBOL programs define the file as **ORGANIZATION IS RELATIVE**. In TIP/ix the file is defined as **DAM** or **DMIRAM**.

The file naming conventions used here are consistent. The file name is used as supplied and no extensions are added.

For Micro Focus COBOL a file of **n** byte records is a series of **n + 1** byte record slots. A line feed after the record indicates that the data record is valid. A null byte after the record indicates that the data record has been deleted. In TIP/ix this is the default (**tipfcs**) structure used for DAM and DMIRAM file types.

TIP/ix System Configuration File - tipix.conf

Purpose of tipix.conf

The TIP/ix system configuration file, ***tipix.conf***, controls the operation of TIP/ix. It contains statements to set:

- UNIX environment variables (especially PATH and TZ).
- TIP/ix system parameters,
- TIP/ix log parameters, and

When you start TIP/ix (by executing **tipctl boot**), it looks for *tipix.conf* in the *$TIPROOT/conf* directory. There is only one *tipix.conf* file for each TIP/ix system (LOCAP).

The statements in *tipix.conf* determine the UNIX environment of the TIP/ix monitor, **tipmon**, and the processes it initiates (such as tipfcs). TIPROOT is the only environment variable inherited from the user that starts TIP/ix (with tipctl). All other environment variables and system settings are derived from statements in *tipix.conf*. This ensures that TIP/ix will behave in a consistent fashion regardless of who starts the system.

A sample tipix.conf file is given later in this book.

In addition, each TIP/ix user may optionally have a ***.tipixrc*** file which sets Unix environment variables and runs commands (but does *not* contain system parameters). For details, see the chapter on ***TIP/ix User Run Control Files*** in this book.

Format of tipix.conf

The *tipix.conf* file is a text file containing:

- PARAM statements
- Statements to set environment variables, such as PATH and TZ
- Comments (which start with # or *)
- Blank lines
- [option to test for] This can be a file pathname, or a unix hostname. If the first character is '!' then the test is negated. If a file path is defined and then file path exists the test is TRUE. If a hostname is given and

the current Unix/Linux ‘hostname’ matches exactly then the test is TRUE.

- [:ELSE] defines the else leg of the option tested is false
- [:END] terminates statements to be processed based on the option tested

Sample tipix.conf File

This is a sample *tipix.conf* file for a TIP/ix system:

```
# Sample tipix.conf file
#
# Path for TIP/ix transaction binaries
PATH=$TIPROOT/bin:/usr/local/tip/bin
# Timezone
TZ=EST
#
# Note that there is a difference between
# the environment variable TIPPRINTLPP and the
# param variable PRINTLPP.
#
# The TIPPRINTLPP in the tipix.conf file
# can be overriden by a user's
# TIPPRINTLPP environment variable.
# The PRINTLPP param variable
# can be overridden by the user's TIPPRINTLPP environment
# variable (but not by the PRINTLPP environment
# variable).
#
# Some environment variables that apply to
# TIPFORKed transactions etc.:
TIPPRINT=PS
TIPPRINTLPP=64
TIPPRINTAUX=LP
#
# TIP/ix parameters
PARAM MONLOG=c
PARAM QUELOG=a,10K
PARAM MCSLOG=c,6K
PARAM MCSCACHE=48K
PARAM QBLFILE=/home/data/qbl1
PARAM QBLFILE=/work/data/qbl2
PARAM DTPHEARTBEAT=30
[testcomp.mydomain.com]
PARAM LOCAP=TEST1
INCLUDE test.conf
[:ELSE]
PARAM LOCAP=PROD
INCLUDE prod.conf
[:END]
```

Environment Variables in tipix.conf

Any environement variable required for any software running under control of TIP/ix may be defined in tipix.conf. The possibilities are not limited to but may include the following environment variables set in *tipix.conf*.

Variable	Description
PATH	The PATH should include $TIPROOT/bin, $TIPROOT/scripts, and any other directories containing TIP/ix transactions. This path is searched when TIPFORK is performed on a transaction that has a program definition with a relative path for the executable.
TIPISAMDAT	When set to N will not add the ".dat" extension to ISAM data files created by TIP/ix. Although using ".dat" is C-ISAM compatible, Micro Focus COBOL does not append the ".dat". i.e. TIPISAMDAT = N
TQLSYNC	When a TQL program is initiated TQL will search for this environment variable. If it the value of this variable is "on" then SYNC will be processed and will align data items to the SYNC boundaries.
TZ	Some system behavior is based on the time zone. For example, Canada and the US use a period as a decimal point, whereas most of Europe uses a comma.
DTPHEARTBEAT	In conf/tipix.conf you may place DTPHEARTBEAT=n where n is the number of minutes to go without hearing from another TIP/ix locap before sending a message to check if it is still there. The default is to never check. The value is given as a number of minutes. PARAM DTPHEARTBEAT=30 Note: All other TIP/ix system must be upgraded to this patch level before using this feature.

Variable	Description
User defined	You may find it convenient to define environment variables in *tipix.conf* to make it easier to specify the location of programs and files with **smprog** and **smfile**. See the description of *Path/Label* in **smprog** and **smfile** in the *TIP/ix Utilities*.

Note: If you use Micro Focus COBOL in a dynamic linking environment and either:

- background transactions, or
- IMS programs with REUSE option

then the *tipix.conf* file *must* contain an COBDIR entry.

For other environment variables, see the chapter on *TIP/ix Environment Variables* in this book. For a combined list of TIP/ix, HSP/80, HSP/22, and other related environment variables, browse the file **$TIPROOT/arm/scripts/arm.tipsetenv**.

TIP/ix Parameters

TIP/ix Parameters

The TIP/ix system and log parameters may be declared in the *tipix.conf* file with the PARAM command. For example:

```
PARAM MCSLOG=c,6K
PARAM MCSCACHE=48K
PARAM BACK=10
```

The log parameters are given after the following section.

System Parameters

The system parameters are described below: Defaults are in bold.

ALERT=me@mydomain.com
: If there is a serious problem, like TIPFCS aborts or DTP aborts, or the system is getting low on memory, then an email would be sent to the address defined. Of course your Unix system must be able to send email.

BACK=*n*
: The maximum number of TIP background transactions allowed at one time. The default is **(maxusers/10)+1**. Maxusers is the number of licensed users.

BACKNNN=Y

to have background processes given the USER-ID names of BACK$001, BACK$002, etc. The default is that the background process gets the same User-id as the process which did the TIPFORK.

BACKSECUR=Y

to have background to have background processes take security from the USER-ID names of BACK$001, BACK$002, etc. The default is that the background process inherits security settings from the user whos transaction did the TIPFORK.

CDAsize=*n*

Largest typical size of the CDA (the largest size that is often passed from one transaction to another). TIP/ix will preallocate enough memory to pass a CDA of the size defined with this parameter. The default is **4096**.
If you pass a larger CDA, TIP/ix will allocate the memory as required.

COLORST=*attr*,*fg*,*bg*

Status line attribute and colors.

attr Attribute can be one of:

B	Blinking.
N	Normal (the default).
L	Low intensity.
BN	Blinking and normal.
BL	Blinking and low intensity

fg Foreground color may be one of:

WH	White
CY	Cyan
MA	Magenta
BU	Blue
YE	Yellow
GR	Green
RE	Red
BL	Black

bg Background color may be one of the above colors.

COLORNRM=*attr*,*fg*,*bg*

Normal attribute and colors.

COLORREV=attr,fg,bg

Reverse attribute and colors.

COLORLOW=attr,fg,bg

Low intensity attribute and colors.

COLORBLK=attr,fg,bg

Blinking attribute and colors.

COLORPNM=attr,fg,bg

Protected normal attribute and colors.

COLORPLW=attr,fg,bg
Protected low intensity attribute and colors.

COLORPBL=attr,fg,bg
Protected blinking attribute and colors.

COLOROFF=attr,fg,bg
Off display. To hide information, such as passwords, set the foreground and background colors the same.

COMBINDLOGS=Y|N
Default is Y. If N, then each log file gets a unique name based on date and time.
Note: For this to work properly you must leave the Log file level field in smprog blank. If an "A" or "M" is in the field the log will only have a date stamp.

CONSOLE=path
Specify the device name, path, of the console device. This could also be a file name where TIP/ix will write console messages.
The default is '/dev/console'.

CSMON=MANUAL
Specifies that you want to manually control the ‘connection server’. See the section on ‘Connection Server’ for more details. The default for this option is AUTOMATIC which means TIP/ix will start/stop the connection server.

DBILOG=logoptions
Specifies the TIP/dbi logging options for the TIP/dbi server processes.

DECIMAL=
The default depends on your TZ (time zone). Briefly, North America gets DECIMAL, Europe gets COMMA.

DECIMAL In most English-speaking countries a period is used as the DECIMAL point.
For example: 1234.56
COMMA In most European countries, a COMMA is used instead of the period.
For example: 1234,56

DEADLOCK=DUMP
this tells TIP/ix that on any deadlock status it should send a message to the user's terminal (if present) and then do a TIPDUMP and abort the program.

The default behaviour is to return the PIB-DEADLOCK-DETECTED status and let the program decide what to do about it. When a deadlock has been detected the transaction has already been aborted and rolled back.

DMPPACKOUT=n

Defines how many output messages before which the ACK of the last DMPP-2020 input message should be sent. ACK is sent just before the n^{th} output. Default is 1.

DMPPACKTIME=seconds

Defines how many seconds before the last DMPP-2020 input message must be acknowledged. Default is 3.

DMPPHEARTBEAT=minutes

Defines how many minutes of inactivity before sending a Heart Beat message. The method is often needed to keep the TCP/IP socket alive. The default is no heartbeat.

DMPPOFF=minutes

Defines how many minutes to wait for the next transaction. If nothing arrives within that time, then the socket is closed and waits for the message switch to connect back for more work. DMPPOFF would not be used if DMPPHEARTBEAT is defined.

DPSBACKGROUND=color

Defines default screen color for background of DPS screens. Options are
The default is Ebony.

DPSFOREGROUND=color

Defines default screen color for foreground of DPS screens. Options are
The default is Yellow.

DTPCORE= Yes | **No**

Should a core file be created if tipdtp aborts?
The default is **No**.

DTPsessions=*n*

The maximum number of distributed transactions allowed at one time. The default is **(maxusers/10)+1**
The maximum is 10,000.

DTPHEARTBEAT=n

where n is the number of minutes to go without hearing from another TIP/ix locap before sending a message to check if it is still there. The default is to never check. The value is given as a number of minutes.

DTPRETRY=*n*

Amount of time (in minutes) to wait before retrying a failed connection attempt.
The default, **0**, means "**don't retry**".
Avoid small values. If the PepGate or System 80 is actually down for some reason then TIP/ix could consume a lot of

resources trying to establish a connection that can not be done.
Setting the value at 5 would cut the wait time in half. It usually takes a minute to connect anyway so setting it lower than 5 would start to cause TIP/ix to be continually trying to connect. If you have several connections going through the same PepGate this will compound problems. 10 minutes is reasonable.

DTPTIPUSeR= *name* | **DEFAULT**
The default TIP/ix user id to be assigned to incoming distributed transactions. The default is **DEFAULT**.

DTPUNIXUSeR=*name* | **nobody**
The default Unix user id to be assigned to incoming distributed transactions. The default is **nobody**.

DPSPROMPT=NO
This will result in no TIP prompt after a DPS program terminates

DTPONLYONE=TRM1,TRM9
option to prevent more than one of a given terminal session to be started via DTP. If a new session request arrives with the name as an already existing session, then TIP/ix will purge the existing session and then open the new session.

FCSioFiles=*n*
The maximum number of data files to be managed by a single TIPFCS I/O process. The default is **(maxopenfiles-8)/2**.
The minimum is 5. The maximum is 32.

FCSMATCH=YES | NO

YES When trying to access files that do not exist, TIP/ix will *not* automatically create them.
If the physical characteristics of the file do not match the smfile entry, TIP/ix does *not* open the file, but gives an error message instead.

NO When trying to access files that do not exist, TIP/ix automatically creates them. This is the default and the historical behavior. TIPFCS will adapt to any changes in the real file definition.

FCSPERMission=**ALL** | TIP | READ

ALL Any data file created by TIP/ix will have read/write permission for all users.
The default is **ALL**.

TIP If TIP, any data file created by TIP/ix will only have read/write permission for the TIP/ix administrator user and group.

READ Any data file created by TIP/ix will have read/write for the TIP/ix administrator user and group. Other users get read permission.

FcsOPEN=*[*]listoffiles*

Specify list of files to be opened (marked available for online use) before starting TIP/ix.

FCSQueues=*n*

The number of Unix message queues to be reserved for the TIP/ix I/O processes. The minimum is 10. The default is **20**.
Each FCS server process needs a queue for itself. They are not shared.

FCSREBUILDINDEX=YES

This will cause TIPFCS to have D-ISAM check&rebuild the index if an ENOCURR status is reported on the read&lock which is done after a new record is added to the ISAM file. The ENOCURR status is likely an indication that the index has been corrupted.

FCSCHECKINDEX=YES

This will cause TIPFCS to have D-ISAM verify the index whenever a record is updated and a key field was changed, or a record was added or deleted. If the index is corrupt then the index will be rebuilt. The rebuild is only done if TIP/ix has the file open for Exclusive use. A message will be written to the HISTORYfile if anything was done.

FCSCHECKINDEX=YES,n

The 'n' is the interval to check index. Every 'n' record updates it will check the index. The default interval is 50.

FCSCHECKINDEX=YES,n,REOPEN

The 'REOPEN' parameter tells TIPFCS to close the ISAM file not defined to have exclusive use to TIP/ix and then reopen them as exclusive in order to rebuild the corrupted index. If the re-open fails a message is written to the

HISTORYfile and the file is unavailable until a person solves the problem. If you also had PARAM ALERT=email defined then a message is sent to that email address.

FCSSYNC=

NEVER
: Files are never synched to disk.

MIN
: Updated files are synched to disk at each transaction end point.

MAX
: Sync all I/O to disk as it occurs. This is the default.

FCSWAIT= ***n***

Where ***n*** is the number of seconds to wait before returning PIB-HELD. The default is 1.

GDAsize=*n*

The size of the global data area (GDA). The number you specify is rounded up to the nearest multiple of 256. The maximum is 999,936. The default is **2048**.
Also defined with tipinstall –G.

Note: Placing GDAsize in tipix.conf without further action does nothing. The value specified is not used during TIP/ix startup. GDAsize is effected if you ran tipinstall -L 0.

ILLTRAN=*name*

The name of the transaction to be run when an invalid transaction code is entered.
The default action is to **display an error message**.

HOSTID=*X*

Where X is a letter between A thru Z. The value is returned via a CALL “TIPHOSTID” USING myid.

HOSTTYPE=*X*

Where X is a letter between A thru Z. The value is returned via a CALL “TIPHOSTTYPE” USING mytype.

IMS3D=

Yes
: When using IMS transactions with TIP/fe, display the UTS messages with a 3D effect.

No
: Display in flat mode. The default.

IMSCENTURY=

Yes
: Use 4-digit year in the IMS IMA and PIB YEAR fields. The default.

No
: Use 2-digit year.

IMSDYNOMA=

Yes Send the formatted message to the terminal immediately. This is the default.

No IMS CALL “BUILD” should return the UTS data stream into the OMA.

IMSFORKW=

Yes Use "Output for Input Queuing" printing when using TIP/fe for Windows in Remote MCS mode. For more information, see "*Output for Input Queuing, from IMS Programs*" in the *TIP/ix Programming Reference*.

No The default.

IMSKEEPMCS=

Yes IMS programs will remain in MCS mode (i.e. not read by UTS input) if SFS-TYPE is set to “R” and SFS-LOCATION was cleared. Normally SFS-LOCATION should not be cleared. This may be required by IMS/1100 programs using DPS

IMSMAXRETRY= ***n***

Where ***n*** is the number of attempts to try and lock a record before giving up. If defined a message “Input in process” will be sent to the terminal during retry.

IMSO4IQUE =

Yes The default is that output for input messages are queued until the terminal reaches an idle state and then they are processed.

No At this setting this parameter will prevent output for input messages from being queued. If this is specified, then if the terminal is already busy a status of 6,2 will be returned.

IMSTRENearly=

Yes The IMS API module will issue the TREN (transaction end processing) before sending the OMA back to the terminal on External and Normal termination.
This option will cause the commit/rollback to be processed early and hence release the record locks and database servers much sooner. This would improve the overall system throughput, although it would only be noticeable on busy systems with many terminals.

No The default is to send the OMA and then the TREN is done. The default behavior is more correct since

if for any reason the output message could not be delivered then the transaction would be aborted.

JouRNal=

Yes Enable journaling. The default.
No Disable journaling.

JRNFILE=*jrnfile*
The name of a system journal file. You may specify this parameter twice to get (a maximum of) two journal files. The default is **TIPROOT/tipfiles/tipix.jrn0.**

JRNLGOF=

Yes Write LOGOFF journal records.
No The default is no LOGOFF journal records.

JRNLGON=

Yes Write LOGON journal records.
No The default is no LOGON journal records.

JRNCHECK=*minutes*
Defines how many 'minutes' to go without swapping the journal file. The journal file would be swapped when it reaches its max size defined by JRNSIZE or if it has not been swapped for the defined number of 'minutes'.

JRNSIZE=*n*K
If dual journaling is configured, this is the maximum size of the journal file in kilobytes before TIP/ix swaps to the alternate journal file.
The default, **0**, means that TIP/ix will *not* perform automatic swapping. You must perform any swapping manually (by running **JRNSWAP**).

JRNSWAPSCRIPT=*swapscript*
The name of the script that TIP/ix will run after swapping the journal files. The default name is **$TIPROOT/scripts/tipjrnsw**.
We do not provide a computer-readable script. However, see the **JRNSWAP** utility in the *TIP/ix Utilities* for a sample tipjrnsw script.

JRNSYS=YES
Causes updates to the TIP$SYS control file to be journaled. Normally they are not.

LANGuage=**x**

Declare the language code. The valid language codes are as follows:

A	American English
B	Canadian English
C	Canadian French
D	Dutch
E	UK English
F	French
G	German
H	Swiss-German
I	Italian
K	Afrikaans
N	Norwegian
P	Portuguese
R	Swiss-French
S	Spanish
W	Swedish
X	Danish
Y	Finnish

LDAP=YES

The TIP/ix system may do user authentication using LDAP. You may need to install the optional LDAP client software for the version of Unix/Linux being used.

LOCAP=*name*

The name to be used for this TIP/ix system within the TIP network. This parameter must be specified if distributed transaction processing (DTP) is installed.
Otherwise, the default is the UNIX hostname in uppercase.

LOGIN=UNIX

For INT1 and TIP/ws Direct connect, the user must be authenticated, if you want user authentication to be done using the Unix passwords.

LOGIN=LDAP

For INT1 and TIP/ws Direct connect, the user must be authenticated, if you want user authentication to be done using LDAP. You may need to install the optional LDAP client software for the version of Unix/Linux being used.

LOGINFORM=mcsform,F

For INT1, the MCs format ‘mcsform’ is displayed to solicit the end users userid and password. If the ‘,F’ is declared then the reply to the format is read as basic UTS message and parsed. If the ‘,F’ is not used then the MCS format is read as a form and expected to have a field for the 8 character user id and another field for the password.

LOGDIRECTORY=/location/to/put/logs

When included in *tipix.conf* all log files, for programs used in TIP/ix which have logging feature turned on, are saved in specified directory.
This logging feature has a low precedence in comparison to the environment variable TIPLOGDIR.

LOGEXPIRE=n[H|D]

define how long to keep inactive TIP/dbi log files. For example, LOGEXPIRE=36H keeps for 36 hours, 3D keeps for 3 days. The default is 2 days.

MaXCaLLs=*n*

The maximum number of CALLs a transaction may make before TIP considers it to be in a loop and aborts it. The counter is reset for each input message and CALL TIPTIMER.
The default, **0**, means that **no limit** is placed on the number of CALLs.

MAXFILELOCK=*n*

This will retry *n* times to complete rollback before a system shutdown.

MAXFILESIZE=unlimited or nnnM

Defines the maximum allowed file size for TIP/ix internal processes. Default is ‘unlmited’.

MAXRCSZ=*n*

The size of the largest data record used in the system. The minimum is 2816. The default is **4096**.

MAXTIME=n

After the defined number of seconds of a transaction program running time, the transaction would be aborted.

MCSBUFSIZE=n

n is the size of the MCS staging buffer this defines the maximum MCS Area. Default is 5120

MCSioCACHE=*n*K

The number of Kilobytes to reserve for caching MCS formats. The minimum is 8K. The default is **64K**.

MCSDATE=Yes

When enabled will cause MCS to display embedded date fields using a 4-digit year including century. as follows:
$YYMMDD$ as YYYYMMDD,
$MMDDYY$ as MMDDYYYY, and
$DDMMMYY$ as DDMMMYYYY.
TIP/fe build 283 or later.

MCSDPS=YES

If using DPS then the Windows TFD should be used for screen maintenance and not the UNIX version of TFD. To avoid overwriting the file by mistake.

MCSLONGDATE=YES

will cause MCS to display embedded date field $YYMMDD$ as YYYYMMDD, $MMDDYY$ as MMDDYYY and DDMMMYY$ as DDMMMYYYY. Display 4-digit year including century.

This requires TIP/fe build 283 or later.

MCSLONGTIME=YES

will cause MCS to display embedded the time field $HHMM as HH:MM:SS providing there is space on the screen. This requires TIP/fe 2.4 build 3 or later to work. Older version of TIP/fe will function, but not handle the long TIME display.

MCSNEGZERO=YES

will cause MCS to display the negative edit sign for numeric edited field that hold a value of negative ZERO. In March 1998, MCS was changed to not display negative zero, this option allows you to revert back to the old style handling of this condition. This option also needs the version of TIP/fe to be at least 2.3.5.

MEMCHECK=

Yes Perform frequent validation of user memory areas. ***This is strictly used for debugging***, and should be done in coordination with Inglenet support. If a user's memory area is found to contain invalid data the transaction will be aborted and the memory area will be dumped in the tipmon or transaction log files. Please contact Inglenet should this occur. ***Setting this parameter will degrade system performance***.

No The default.

MONCHECK=*n*

The system will check for aborted TIP/ix command processors every *n* seconds. The default is **180** seconds (3 minutes).

MSGROW=*n*

Specify the row, *n*, on the terminal where unsolicited messages (sent with the MSG and APB commands) will be displayed. If not specified, unsolicited messages have no positioning and just paint wherever the cursor is resting.

NumGRPS=*n*

The maximum number of groups a user may have to be searched. The default is **16**.

OPENFILES =

YES If parameter is set to YES in tipix.conf file. Upon start up of TIP/ix, parameter will open all the files set to “Startup: Open: YES” in smfile.

PCSINCrease=*n*

Specify the amount, *n*, to increase "maximum threads" when PCSMAXwaiting is reached.
The default value is 2.

PCSGROUPSEARCH=YES

This feature will allow you to have different users be connected to different instances of the database based on the user's TIP/ix group membership. By default this feature is disabled. New feature for 'tippcstm' (Transaction and Database interface thread manager) to search its environment by prefixing the user's group name(s) to the schema name and if present, then set the environment of the TIP/dbi II server process to <schema>_xxx from <group>_<schema>_xxx where 'xxx' is any of CON, DSN, SID, UID, PWD.

PCSMAXwaiting=*n*

Specify the maximum length, *n*, of the wait queue for a TIP/dbi server or re-usable transaction. If the number of waiting requests exceeds this value, then the "maximum number of threads" is increased.

PCSRELOADENV=YES

New feature to have '`tippcstm`' completely reload its environment from the tipix.conf whenever a 'tipctl ping' is issued. Normally only the PARAM statements get processed but with the following option added to tipix.conf, `tippcstm` will now also re-establish its environment.

PCSstack=*n*

The maximum program control system (PCS) stack level. This is the maximum nesting of CALL TIPSUB. The default is **16**.

PREVENTDEBUG=YES

This will turn on any ‘debug’ flags set for a transaction. This might be something to use on a production only system where you do not want any program to accidentally stall waiting for the debugger to connect.

PrintLF=

Establishes the default TIPPRINT option for appending a line feed to each message transmitted to an auxiliary device. TIPPRINT normally buffers a number of output print lines so that a screen full of data can be sent to the terminal and printer as one output message.
If an online program does not specify a value for the PRINT-LINE-FEED field when the TIPPRINT interface is opened, this specification (possibly overridden by an **smterm** definition) is taken as the default by TIPPRINT.

No Specify this (default) value so that TIPPRINT ensures there is NO Line Feed (LF) character at the end of a screen full of data sent to a printer. In this case, the terminal emulation software must provide a LF character.

Yes Specify this value in situations where a LF character is required at the end of each output screen.

Note: If this value is incorrectly specified, reports printed on auxiliary printers may appear with an occasional extra blank line or a missing blank line.
Specify this keyword as required by the majority of auxiliary printers in use at your site and override the specification with **smterm** definitions for the terminals that require a different setting.

PrintLPP=n

Specify the default number of lines per page for an auxiliary device.
If an online program does not specify a value for the PRINT-PAGE-LEN field when the TIPPRINT interface is opened, this specification (possibly overridden by a **smterm** definition) is taken as the default by TIPPRINT.
Default: PRINTLPP=60

PrintLFFF=

Specify the default action for TIPPRINT when printing form feed characters.

Yes The default is to precede the form feed with a linefeed.

No Don’t precede the form feed with a linefeed.

PrintTOF=

Specify the default action TIPPRINT should take before and after printing to an auxiliary printer.
If an online program does not specify a value for the PRINT-TOP-OF-FORM field when the TIPPRINT interface is opened, this specification is taken as the default by TIPPRINT.

Yes Specify YES to ensure that a form feed is issued when an FCS-OPEN function or an FCS-CLOSE function is performed.
If the first (or last) output via TIPPRINT already causes a form feed, TIPPRINT does not insert an extra one.

Top Specify Top to ensure that the first output is a skip to top-of-form and that there is NO skip to top-of-form at the ending of output. (If the very last line in the output stream is a skip to top-of-form, TIPPRINT will actually remove it.)

Bot Specify Bottom to ensure that there is NO skip to top-of-form at the beginning of output and that there IS a skip to top-of-form at the ending of output.

No Specify NO to ensure that there is no skip to top-of-form at the beginning of output and no skip to top of form at the ending of output. This is the default.

PrintTTL=

Specify TIPPRINT's default action for title pages.
If a user online program does not specify a value for the PRINT-TITLE field when the TIPPRINT interface is opened, TIPPRINT uses PRintTTL to determine if title pages are required.

YES The default is to print title pages.

NO Don't.

PrintUC= YES | **NO**

Specify TIPPRINT's default action for converting alphabetic characters in a print line to upper case.
If an online program does not specify a value for the PRINT-UPPER-CASE field when he TIPPRINT interface is opened, this specification is taken as the default by TIPPRINT.
Default: PRINTUC=NO

ProMPT=*String*

Specify string to be used in the TIP/ix command processor prompt.

TIP/ix The default string is **TIP/ix**.

LOCAP

Use the local “LOCAP” name in the TIP/ix command processor prompt.

QBLCHecK=*n*

Check the size of the Quick Before Look (QBL) files every *n* minutes. If they are large, TIP/ix pauses the system for a few seconds and packs the QBL files.

QBLFILE=*qblfile*

The name of a system QBL file. You may specify this parameter up to six times to get multiple QBL files.
To get an advantage from having multiple files, each file must be physically on its own spindle (physical device).
The default is **TIPROOT/tipfiles/tipix.qbl0.**

QBLMAXSiZe=n

Specify the size *n* in kilobytes (K) or megabytes (M) at which the QBL file should be packed. The default is **3M**. This does not cause a qbl pack operation by itself, but is used in conjunction with the QBLCHecK parameter.

QUEUEUNSOL=NO

Declares that message being sent unsolicited to a terminal which is not connected should be discarded. This is the default behaviour.

QUEUEUNSOL=YES

Declares that message being sent unsolicited to a terminal which is not connected should be queued on disk until the terminal connects. When TIP/ix starts all previously queued message will be discarded.

QUEUEUNSOL=/directory/tosave

Declares that message being sent unsolicited to a terminal which is not connected should be queued on disk in the named directory until the terminal connects. When TIP/ix starts all previously queued message found in this directory will be retained as queued.

QUEUEUNSOLTO=(*prefix,*prefix2,...,*prefix8)

A list of up to 8 prefixes of terminal names that would have unsolicited messages queued for. If an unsold msg is sent to a terminal name not matching this list, then it is NOT queued on disk for later processing.

Example:

PARAM QUEUEUNSOLTO=(*WM,*WN,*WQ,SPH1,*T91)

QUEueINTerval=*n*

Schedule the TIPQUEUE server every *n* seconds, if there is any data in the queue. The default is **60**.
The value specified here is a default value that is only used if the interval is not defined in the queue definition.

QUEueRESta rt=*n*

The time in seconds to wait before restarting the transfer of TIPQUEUE messages to another TIP system. The default is **300**.
The value specified here is a default value that is only used if it is not defined in the queue definition

RESTART=YES

attempt restart of system if it fails

RESPLOG=*n*

Take a snapshot of the system response time every *n* minutes. The response-time log is kept for the last 24 hours. It may be displayed with the "**status** T" command. The default is 15 minutes.

SAVELOGS=directory

Defines where the log files will get saved into a compressed tar file. (Do NOT make a save directory which is a sub-directory of $TIPROOT/log.) The log files get saved each time TIP/ix starts up.

SAVELOGEXPIRE=n[H|D]

to define how long to keep log archives created with the SAVELOGS parameter. (Default is 6 months) If you want to keep the log archives forever then you must specify PARAM SAVELOGEXPIRE=0.

SESSIONCONF=YES|NO

Tells 'tipix' to process the $TIPROOT/conf/session.conf control file. This can be used to turn tipix and/or application logging on or off. Default: Yes

SFSFILL = "x"

Can be declared in the tipix.conf file to set the MCS FILLER value, where "x" may be "*", "_", or " ". The default is "_".

SIGNEBCDIC = YES

Tells TIP/ix that the applications are compiled with SIGN=EBCDIC for Micro Focus or –fsign-ebcdic for OpenCOBOL/COBOL-IT.

SMSEMULATOR = YES

When this parameter is set to YES then SMS/90 calls are supported, this will set the support for SMS handling system wide. The individual programs which need to

participate must have their catalogue (SMPROG) entries updated to state that their program type is SMS (not IMS). All this assumes is that the screens have been converted on the System/80 with the latest version of the SMSCNV utility, and brought over to Unix.

SMSFILL =“x”

Can be declared in the tipix.conf file to set the MCS FILLER value, where “x” may be “*”, “_”, or “ “. The default is “_”.

SOAPERROR = myext

Where 'myext' is the name of the template file extention to use for sending back error messages on the SOAP/XML interface. The actual file name will be in $TIPROOT/tipsoa and called default.myext. You can pick any name you like for 'myext' and make the file have what you want. The default is “error”.

TERMUNATT= <terminal type>

Use the TERMUNATT PARAM to specify the value for the TERM environment variable when a tty-mode unattended terminal is defined in SMTERM.

Example:

If a DEC-VT220 terminal is attached to /dev/tty7 and an SMTERM entry defines that device as "Unattended? Y" then any output messages which are directed to that terminal will require that a "tipixtty" UI process be started on that device. In order for "tipixtty" to correctly display any output (i.e. using curses/terminfo) the TERM environment variable must be set. "tipmon" will use the value of the TERMUNATT parameter to set the TERM environment variable for the new "tipixtty" process.

TIMeoff=*n*

The maximum number of *minutes* a user may sit at the TIP/ix system prompt without running a transaction. After this time the user is automatically logged off TIP/ix.
The default is **14,400** minutes (10 days).

TIMeouT=*n*

The maximum number of *minutes* a transaction may remain in external succession before being aborted due to time out. After the time has expired a program doing a TIPMSGI or TIPTERM GET will be given control with the PIB-TIMED-OUT status set.
The default is **14,400** minutes (10 days).

TIMeouT=*n*,**LOGOFF**

The maximum number of *minutes* a transaction may remain in external succession before being aborted due to time out. After the time has expired a program doing a TIPMSGI or TIPTERM GET will be aborted and the user session logged off.
The default is **14,400** minutes (10 days).

TIMEIDLE=*n*

The maximum number of *minutes* of idle time before sending a ping to TIP/ws or TIP/fe. This may be helpful to keep the network connection active and avoid network timeout. The default is that a 'ping' is never sent.

TIPDMSRESEQ = n

This variable may be used to set the increment value for TIP/dbi . You may want to set this if you know your program will be doing a lot of inserts, "n" can be in the range 2 to 16.

TIPINT1=YES

to support UTS INT1 DCP style over TCP/IP protocol. TIP/ix will listen on the TCP/IP port 256 for connections using this protocol. You may also specify the TCP/IP port number before the option YES.

(Eg TIPINT1=102,256,YES)

TIPINT1=LOGIN

If you want INT1 users to have to logon with a format

TIPINT1=DEMAND

If you want INT1 users to logon with a command line style

TIPINT1=102,HLC

If you want INT1 users to connect via HLC (host LAN controller) protocol

TIPWS=port

Defines the TCP/IP port TIP/ix should listen on for TIP/ws to connect directly to TIP/ix. You may choose any port which is not already used as long as TIP/ws is defined to use the same port.

TMPDIR = /*directory_name*

A User can specify what directory TIP/ix uses when updating TQL programs or records. The default is ***/var/tmp***, a temporary directory is used so that the original is not altered until a working alteration is achieved.

TRANSUSPEND=maxseconds

Defines the maxmum number of seconds that a suspended transaction (via CALL TIPSUSPEND) will be held. After

that time period the transaction is rolled back. The default value is 15.

TRANSUSPENDCDA=YES
Tells TIP/ix to save the CDA on a TIPSUSPEND PUT and restore the CDA on a TIPSUSPEND GET. This allows the application to retain some context data with the suspended transaction. Default is NO. (Although it seems like a good idea to use the YES option.)

UASERVICE=port[,options]
'port' or 'ports' is the TCP/IP port that TIP/ix is to listen on for UniAccess ODBC connections and messages. Only RPC calls are supported. You can define 1 to 3 different port numbers separated by commas.

Possible 'options':

single – limits one ODBC transaction at a time

multi – starts as many ODBC transactions as arrive

Default: multi

TIP/ix also supports RPC calls via the Windows SQL Server ODBC client driver software.

UAUNIXUSER=usr
Unix User-id to be assigned to UniAccess transactions default is 'nobody'

UATIPUSER=usr
TIP/ix User-id to be assigned to UniAccess transactions default is 'WEBDFLT'

UNIXGROUPS=*name,name2,…name8*
Define up to 8 additional Unix groups which TIP/ix should make itself a member of. You may need TIP to be a member of the Oracle DBA group or some local application group.

USeRQueues=*n*
The number of Unix message queues to be reserved and shared among the TIP/ix user interface processes. The default is **10**.

USERS=*n* Maximum number of interactive users allowed. The default is **maxusers** (the number of licensed users.)
This value may be less than or equal to that allowed by the TIP/ix software activation key.

UTSKEYS=

Set global default for insert and delete keys.

yes The insert and delete keys work as they would on a UTS terminal.

no The insert and delete keys work as they would on a computer. See **smuser** for details.

WEBSERVICE=port[,options]

'port' or 'ports' is the TCP/IP port that TIP/ix is to listen on for SOAP/XML connections and messages. You can define 1 to 3 different port numbers separated by commas.

Possible 'options':

single – limits one SOAP transaction at a time

multi – starts as many SOAP transactions as arrive

Default: multi

WEBSESSIONS=n

n is maximum web sessions allowed to be active Is limited by the maximum users in the TIP/ix license

WEBALTER=(bgn,end,num,bgn,end,num)

where bgn & end specify time of day in hh:mm format. If only two digits are given, they are assumed to specify the hh value and the mm value will be zero. For example 08 is 8:00AM, 17 is 5:00PM, etc. If the current time of day is between bgn & end, then WEBSESSIONS is altered to be n. You may supply 1 or 2 sets of alternate values.

WEBTIMEOUT=n

n is minutes of idle time before killing session default is 120 minutes

WEBUNIXUSER=usr

Unix User-id to be assigned to web users default is 'nobody'

WEBTIPUSER=usr

TIP/ix User-id to be assigned to web users default is 'WEBDFLT'

WEBMCSBUFSIZE=n

is the size to be reserved for receiving SOAP/XML messages. This defaults to 32K. Keep it under 64K.

WEBPREFIX=x

1 or 2 letters used to prefix a unique terminal name for TIP/ic sessions. Default is 'W'.

XMLPREFIX=x

1 or 2 letters used to prefix a unique terminal name for Web Service sessions. Default is 'X'.

XMITkey=*n*
: Declares that a function key is to be used as the "TRANSMIT" key. This may be a value between 1 and 22. By default, it is **not set**.

CITAUNIXUSER=usr
: Unix User-id to be assigned to CITA transactions default is 'nobody'

CITATIPUSER=usr
: TIP/ix User-id to be assigned to CITA transaction default is 'WEBDFLT'

SOAP Parameters

There are many parameters for the operation of the SOAP interface documented in a separate document.

Log Parameters

Some of the PARAM statements define what information is to be recorded in a log file. The parameter value is a series of options separated by commas. The order of the options is not important.

*n*K
: Do not let the log file exceed *n*K bytes in size. Any numeric value may precede the K. The default is that the file will grow as needed.

*n*M
: Do not let the log file exceed *n* megabytes in size. Any numeric value may precede the M. The default is that the file will grow as needed.

a
: Log all commands, warnings and details.

c
: Log commands.

d
: Log details about the command.

t
: Record internal software trace/debug information. (This information may be requested by Inglenet customer support.)

w
: Log warnings.

f
: Turn logging off.

k
: Retain log file after system shuts down gracefully.

n
: Turn logging on.

o=filename
: To specify the output log file name.

Log details of SQL commands issued.

r Log details of records processed.

s Log statistics.

h Record the time stamp accurate to 1/100 of a second, if possible.

The PARAM statements for controlling the log files are as follows:

PARAM	Log file	Description
DTPLOG	log/tipdtp	Distributed transaction processing manager
FCSLOG	log/tipfcs0	TIPFCS processes
MCSLOG	log/tipmcsio	TIP/ix MCS format cache
MONLOG	log/tipmon	TIP/ix system monitor process
PCSLOG	log/tippcstm	Transaction and database server manager
QUELOG	log/tipque	TIPQUEUE scheduling process

Example:

```
MONLOG=4M,a
```

Note: RESPLOG does not control a log file. See RESPLOG in previous section.

Session Configuration File

In the $TIPROOT/conf directory you may create a special configuration file called '`session.conf`'. This file is used to define options/parameters for connections (terminal sessions) made to TIP/ix via special TCP/IP port such as those used by the DMPP-2020 Law Enforcement Message protocol, TIP/ic or WebTS sessions, or SOAP.XML sessions. This configuration file may have any of the following parameters (comment lines begin with #)"

TERM=name	Define the terminal name
PID=number	Define the specific 2200 PID value to be used
PIDPOOL=begin,number	'begin' is the starting PID value and 'number' defines how many numbers follow. So the range is 'begin' thru 'begin+number'. Each new session is assigned an unused PID value in the defined range.
USER=name	Defines the default user name to use for TIP/ix session
LOG=YES	Turn on TIP/ix User Interface logging for this session

LOGAPP=YES	Turn on Application debug logging for this session. Look for the file called 'terminalname.log' in $TIPROOT/tmpwrk or for WEBTS/TIPic & SOAP look in the WEBUNIXUSER home directory.
APPENDLOG=N	The application log file will be appended to on each use. Declare this parameter if you want the log file over-written on each use.
LOGMSG=YES	For DMPP-2020 and SOAP will cause the text of the input/output messages to be written/appended to a file in $TIPROOT/log called dmpp<term>.log where <term> is the terminal name for DMPP, or xml<term>.log of SOAP/XML sessions.
LOGIN=Y|N	For the given port does a SOAP session require userid & password to authenticate before proceeding.
CONVERSATIONAL=Y|N	For a given port does a SOAP session operate in a conversational (Y) or transactional (N) manner.

CITA interface parameters follow the TCP/IP port in brackets.

Define Input messages	
DLINCHDR=Y|N	Data length includes the header or not
DLLEN=nn	Length of the 'data length' field.
DLOFFSET=nn	Offset to location of 'data length' field. (Zero relative)
DLTYPE=A	The data length field is in ASCII (display) format.
DLTYPE=B	The data length field for input messages is in binary (big-endian) format.
HDRLEN=nn	Length of each message header
Define Output messages	
OUTDLINCHDR=Y|N	Data length includes the header or not
OUTDLLEN=nn	Length of the 'data length' field.
OUTDLOFFSET=nn	Offset to location of 'data length' field. (Zero relative)
OUTDLTYPE=A	The data length field is in ASCII (display) format.
OUTDLTYPE=B	The data length field for input messages is in binary (big-endian) format.
OUTHDRLEN=nn	Length of each message header

Each unique TCP/IP port may have unique parameters following a section header such as [nnnn] where 'nnnn' is the TCP/IP port number or another grouping such as 'WebTS', 'SOAP', 'TIP/ic', 'UNIACCESS', 'CITA'. If the value inside the brackets matches then the parameters which follow are processed otherwise they are skipped. You may also specify a file path name to check for the presence of or

a name used to compare to the current Unix hostname, or the Unix userid or if using TIP/fe or TIP/ws the computer name of the end user's PC.

Example:

```
# Uncomment LOG to cause TIP/ix user interface logging
#LOG=Y
# Uncomment LOGAPP to cause application logging
#LOGAPP=Y
# LEMS Terminal options for port 26017
[26017]
TERM=LMDC
PID=558
# Terminal options for port 26018
[26018]
TERM=LSOF
PID=560
LOGAPP=Y
# Terminal options for TIP/ic
[WebTS]
TERM=WEB1
PID=950
# Terminal options for SOAP/XML
[SOAP]
TERM=SOA1
PID=959
[rjn]
LOG=Y
```

The `session.conf` file is only checked for special connections to the TIP/ix system. If you want it to be checked for all end users (even those with interactive terminal sessions) then you can add to `tipix.conf`: PARAM SESSIONCONF=YES

With SESSIONCONF=YES then you might want to turn on certain options for all users ([*ALL]) or just a specific user ([username]) or a specific TCP/IP port that was used to connect to TIP/ix system ([port#]).

Besides the parameter keywords, if a line starts with an '@' then it is taken to be an ECL statement such as @ASG that is to be executed for this session.

If a line starts with SETENV then it is taken to be a set environment variable statement and UNSETENV is to unset an environment variable. Example:

```
@ASG NEBF*NEAP-BAL
@ASG NEBF*NEAP-RATE
[rjnx]
LOG=Y
LOGSAVE=Y
LOGAPP=Y
[SOAP]
LOGSAVE=Y
LOGAPP=Y
LOG=Y
LOGMSG=Y
[8082]
SETENV MYLIB /home/me/bin
[8084]
```

```
SETENV MYLIB /home/you/bin
[7075]
setenv MAX_LOG_SIZE 1900
PIDPOOL=9700,300
HDRLEN=46
DLOFFSET=44
DLLEN=2
DLINCHDR=NO
DLTYPE=BINARY
TCODEOFFSET=0
TCODELEN=6
MAXSIZE=4096
PASSHDR=YES
LOGSAVE=Y
LOGAPP=Y
LOG=Y
LOGMSG=Y
DISABLE=N
```

Share configuration file

In the $TIPROOT/conf directory you may create a configuration file called ‘`share.conf`’. This file is used to define the pre-loading of transaction programs as shared code modules. This is done for performance reasons only. This configuration file may have any of the following parameters (comment lines begin with #)”

LIBPATH=path	Define a single directory to be added to LD_LIBRARY_PATH
MFCSO=	Define the MicroFocus ‘callable shared object’ module
LIB=path	Define an application shared library that should be loaded
LOAD=module.so	Name of a module to be preloaded

The share.conf may also be separated by sections so that some modules are only loaded for specific TCP/IP ports. For example:

```
#Load Shared code for MF COBOL Transactions
LIBPATH=$TIPSITE/lib
LIBPATH=$TIPSITE/bin

[DMPP-2020]
MFCSO=$TIPROOT/lib/lib2200CSO.so
LIB=$TIPSITE/lib/libonlsub.so
LOAD=hst2lj.so
LOAD=lj2hst.so
LOAD=pxcrtj.so
LOAD=mcbout.so
LOAD=gcdmmy.so

[26024]
LOAD=raedit.so

[26020]
LOAD=gp1000.so
```

TIP/ix User Run Control Files - .tipixrc

The *.tipixrc* file may be used to set up the user's Unix environment when they enter TIP/ix. Commands may also be run from the user's *.tipixrc* file.

If desired, users can specify the **tipix** executable as their UNIX shell (rather than **ksh** or **csh)** so that TIP/ix is invoked when the user logs in, and the user is logged off from UNIX when they exit TIP/ix.

In contrast with the TIP/ix system configuration file, the environment specified in the user run control file is *merged* with any existing environment, so the use of *.tipixrc* files is optional.

A sample *.tipixrc* file is given later in this chapter.

Search Order

Before any user can start a TIP/ix session, an administrator must start the TIP/ix system with **tipctl boot**. The TIP/ix system then looks for the system configuration file *tipix.conf* in the *$TIPROOT/conf* directory.

When a user starts a TIP/ix session (by running tipix):

- the user's home directory is searched for a file called *.tipixrc*.
- If this file is not found, and the variable TIPROOT is defined in the user's environment, then TIP/ix searches for a file called *tipixrc* in the directory named by *$TIPROOT/conf*.
- Finally, if this file is not found, or if TIPROOT is not set, TIP/ix will look for the file */etc/tipixrc*.

If the **-f** option is specified to **tipix**, no run control file processing will take place.

Run Control File Format

A TIP/ix run control file, .tipixrc, is a text file containing:

- Commands (which may be given in upper, lower, or mixed case). Each command (after expansion and including line continuations) may be up to 1024 bytes in length.
- Comments (which start with # or *).
- Blank lines.

The following features are supported:

Feature	Code	Description
Continuation	\	To continue a line, end it with a backslash \.
Environment Variable Substitution	$ev ${ev}	The value of the environment variable (ev) is substituted. If the variable does not appear in the environment, the null string is substituted.
Process ID	$$	The special name $$ is replaced with the process ID of the program processing the file (that is, the TIP/ix shell).
Escape Special Chars	\	To code a special character, such as $ or \ precede it with a backslash. Examples: \$ and \\

Sample User Run Control File

This is a sample *.tipixrc* file for a user:

```
# Sample .tipixrc file
#
# This file sets a number of environment variables necessary
# to run TIP/ix.
SETENV TIPROOT=/tipix
# Notice the backslashes used in the following example to split
# a command over several lines:
SETENV PATH=/bin:/usr/bin:/usr/ucb/bin:\
/usr/local/bin:/usr/cobol/bin:\
/tipix/bin:$HOME/tip/bin
# A useful technique is to keep settings common to all users
# in $TIPROOT/tipixrc, and to INCLUDE that file from individual
# users' .tipixrc files.
INCLUDE $TIPROOT/tipixrc
# The following demonstrates the use of the RUN and INCLUDE
# commands to set the TERM variable according to what the user
# types, with a default of "tipfe."
TEMPFILE=/tmp/term.$$
RUN echo 'Enter terminal type (default tipfe): ';\
read ans; echo SETENV TERM=${ans:-tipfe} > $TEMPFILE
INCLUDE $TEMPFILE
RUN rm $TEMPFILE
UNSETENV TEMPFILE
```

Run Control File Commands

A run control file may contain any of the following commands:

Assignment of Variables

Syntax:

```
SETENV VAR=value
SETENV VAR value
```

```
VAR=value
```

Each form of this command will set the environment variable ***VAR*** to *value*. The *value* can contain an environment variable.

Example:

```
PATH=$HOME/tip/bin
```

Removal of Variables

Syntax:

```
UNSETENV VAR
```

This will delete VAR, if present, from the environment.

Removal of All Variables

Syntax:

```
CLEARENV
CLEARENV ALL
```

CLEARENV will remove all variables from the user's environment except HOME, PATH, USER, LOGNAME, MAIL, and TZ.

CLEARENV ALL will remove *all* variables, including these. This is not normally needed, particularly if TIP/ix is invoked directly by the login procedure rather than through one of the UNIX shells (**sh**, **csh**, **ksh**); it is provided so that system administrators can ensure that the only environment variables in the user's initial environment are those specified in the run control file.

Run a Unix Command

Syntax:

```
RUN unix-command
```

This runs the specified *unix-command* using the Bourne shell (**/bin/sh**). This is useful for printing out messages, or running programs which initialize the user's terminal, for example. It can also be used in conjunction with the INCLUDE command to place the output of UNIX commands into the environment.

Note: The RUN command cannot be used to run TIP/ix transactions; only UNIX commands may be run from a run control file.

The RUN command is not permitted in *tipix.conf* files; it may only appear in user *tipixrc* files. Any occurrences of the RUN command in a *tipix.conf* file will be silently ignored.

Include an Alternate Run Control File

Syntax:

```
INCLUDE alternate-rc-file
```

This opens *alternate-rc-file* and processes the commands in it as if they were included in the current run control file. INCLUDE commands may be nested up to 16 levels.
Environment variable substitution is done on *alternate-rc-file*; see the SETENV command for details.

Exit a Run Control File

Syntax:

```
EXIT
EXIT ALL
```

This immediately terminates run control file processing for the current file. EXIT ALL recursively terminates run control processing for the file(s) which INCLUDEd the current one.

TIP/ix System Operation

Once the TIP/ix system is installed you may begin to use it. The first thing you must do is to bring up the TIP/ix system.

TIP/ix is started with the **tipctl boot** command. It is shut down with the **tipctl shutdown** command. To shut down the TIP/ix system, you must be logged on to Unix as:

- a user id that has at least TIP/ix **MAST** level security, or
- Unix **root**, or
- the TIP/ix administrative user id (usually **tipixusr**).

UNIX Daemons

When the system is running a few programs are started as Unix daemons (background programs):

tipmon
: The system monitor.

tipmonldap
: The system monitor which includes an LDAP interface

tipfcs
: The TIP/ix file system. There may be several of these Unix daemons started.

tipdtp
: The TIP/ix distributed transaction processing management process.

tipmcsio
Searches for MCS formats and caches them in memory.

tippcstm
Manages re-usable transactions (usually IMS) and database interface server processes.

tipque
The TIPQUEUE scheduler process.

You can add a "**tipctl boot**" command to the Unix system startup procedures. If you are unable to do this, your Unix system administrator can do this for you.

End users interact with the TIP/ix system and application programs through the **tipix** program. **tipix** manages the terminal interface and interacts with the application programs by sending messages and using system shared memory blocks.

UNIX Kernal Parameters

The following information provides some guidance for the UNIX system administrator in configuring UNIX for TIP/ix. TIP/ix executes as a collection of processes under control of the UNIX operating system.

TIP/ix is available on several different versions of UNIX.

This section uses examples based on UnixWare. The parameter names and utilities for your version of UNIX may vary.

UnixWare
On a UnixWare system the directory **/etc/conf/cf.d** holds a file called **mtune** that defines the boundary values for the kernel parameters. The file **stune** defines the local modifications to the parameters. See the UNIX documentation for **idtune** and **idbuild**.

Other
Most UNIX systems have **idtune** and **idbuild**. In addition, some have menu driven administration utilities: For example:

AIX smit
HPUX sam
SCO sysadmsh

You should read this section in conjunction with the TIP/ix system parameter documentation and your UNIX kernel parameter documentation.

This discussion only provides some guidelines for running a *single TIP/ix system* on UNIX. If you are running more than one TIP/ix system on the same computer or other products (such as a database) that require kernel resources, you will have to make the appropriate adjustments.

Process memory

Oracle recommends that the parameters SDATLIM, HDATLIM, SVMMLIM, HVMMLIM, SFSZLIM, and HFSZLIM all be set the maximum allowable value of 0x7FFFFFFF. These parameters define limits on process memory size. These settings are also acceptable for TIP/ix.

Process memory

Oracle recommends that the parameters SDATLIM, HDATLIM, SVMMLIM, HVMMLIM, SFSZLIM, and HFSZLIM all be set the maximum allowable value of 0x7FFFFFFF. These parameters define limits on process memory size. These settings are also acceptable for TIP/ix.

UNIX shared memory

Some systems require shared memory:

SCO and HPUX
: TIP/ix requires the use of shared memory on SCO and HPUX.

All other
: On all other systems the preferred method is to use memory mapped files.

This section is only relevant for SCO and HPUX.

UNIX shared memory is swapped to the UNIX system swap space which must be large enough. TIP/ix uses two shared memory segments.

TIP/ix System Table Memory

One memory region holds all of the TIP/ix system control tables and is defined by the **tipinstall -M** parameter. The tables allocated and released dynamically from this region are for key holding.

After TIP/ix has been running in active use for a while, run **status S** to determine how much of this memory is used.

If the value for “Most used” is getting close to the value of “-M” then you should increase the -M value. (You can only do this when TIP/ix is shut down.) You should target the “Most used” be about 80% of the -M value to give you lots of room.

Over allocating -M when it is not needed will not improve performance and could make things worse by consuming more memory from UNIX.

TIP/ix User Table Memory

The second memory region is used to hold information for each user connected into TIP/ix. This is defined with the **tipinstall -L** parameter.

Several things affect this value including the number of users, background processes, inbound DTP, maximum data record size, and average CDA size.

When tipctl is used to start up TIP/ix it may report a recommended change to the -L value:

If -L is too small, TIP/ix will stop if it reaches full capacity.

The -L value may be slightly larger than needed. However, if -L is much too big, it does not improve performance and could make things worse by consuming too much memory.

SHMMAX
: the maximum shared memory segment size

SHMMIN
: the minimum shared memory segment size

SHMMNI
: maximum number of shared memory identifiers system wide

SHMSEG
: number of shared memory segments per process

On systems which do not support memory mapped files such as SCO and HPUX you should set the following UNIX kernel parameters to these minimum values.

SHMMAX
: larger of tipinstall -M or -L values plus a couple of MB for future expansion

SHMMIN 512

SHMMNI 100

SHMSEG 15

UNIX semaphores

TIP/ix uses UNIX semaphores to synchronize activities within the system. TIP/ix uses one semaphore set of 15 semaphores. Some data bases like Oracle have larger requirements than TIP/ix.

SEMMNS
: number of semaphores in the system

SEMMNI
: number of semaphore set identifiers system wide

SEMMSL
: maximum number of semaphores per set identifier

The following table list the recommendations provided by Oracle and these are acceptable for TIP/ix.

SEMMNS 200

SEMMNI 70

SEMMSL 25

UNIX message queues

TIP/ix uses UNIX message queues to communicate requests and responses between applications and other TIP/ix system processes. The average size of a message sent over a message queue is about 128 bytes and would not exceed 512.

Message queues can be shared by the TIP/ix user interface processes as defined by the TIP/ix parameter USERQUEUES. A recommended value for USERQUEUES would be the number of TIP/ix users divided by 10.

Each TIPFCS server process has a dedicated message queue as defined by FCSQUEUES. When the TIP/ix system is running the status Q command will report on the usage of messages queues.

MSGMAX
: maximum size of a single message

MSGMNB
: maximum bytes on a single message queue

MSGMNI
: number of message queues system wide

MSGSEG
: maximum number of message segments system wide

MSGSSZ
: size of a message segment

MSGTQL
: maximum number of messages at once system wide

For a single TIP/ix system take the following values as a minimum. (Remember to consider what else is being used on the UNIX system at the same time.)

MSGMAX 1024

MSGMNB
: 256 * (number of TIP/ix users + background + inbound DTP)

MSGMNI
: 6 + USERQUEUES + FCSQUEUES

MSGSEG
: 10 + (number of TIP/ix users + background + inbound DTP)

MSGSSZ 128

MSGTQL
: 10 + (number of TIP/ix users + background + inbound DTP)

Other UNIX parameters

Some other parameters which could affect TIP/ix as mentioned below.

FLCKREC
: number of record locks system wide.

NCALL
: call-out table. Make this at least 10 + (number of TIP/ix users + background + inbound DTP)

NFILE
: how many files can be opened system wide at once

NHINODE
: inode hash table size

NINODE
: controls inode table entries used to manage files

NOFILES
: how many files can one process open. Could affect the TIPFCS servers.

ULIMIT
: controls the size a file may grow to

Tipinstall recommendations

The **tipinstall** utility has a new command line option of **-u**. The option will display some recommendations for minimum values for the related UNIX kernel parameters based on the single TIP/ix system. These recommendations do not take into consideration any other products which may be using the UNIX system. For example:

```
tipinstall -u
```

```
$ tipinstall -u
TIP/ix System installation and initialization
TIP/ix ver 2010/03/15 2.3 R0 - 0176 (c) 1991-2010 Inglenet Business
Solutions
 System files in directory '/u/tipix/' Host node uw7test
Inglenet Business Solutions Registered for the following products on
uw7test
  Concurrent Serial
Product Users Expiry Number
TIP/ix 10 2010/12 20300483
 TIP/ix administrative uid is tipixusr(400), gid is tipixgrp(103)
 Currency symbol character is -C $
```

```
 Decimal point character is -D .
 Activation Area shared memory -L 600K
 Global System shared memory -M 148K
 Global Data Area (GDA) -G 2K
 Memory Type (-TS=shm -TM=mmap) -TM (shared memory file
"/u/tipix/tipmmap")
For this TIP/ix system of
USERS = 10
BACK = 2
DTP = 2
FCSQUEUES = 20
USERQUEUES = 10
CDASIZE = 4096
MAXRCSZ = 4096
Recommended minimum UNIX kernel parameter values
NPROC 130
MAXUP 40
FLCKREC 300
NCALL 30
SEMMNS 200 Is now set at 100 in UNIX
SEMMNI 70
SEMMSL 25
MSGMNB 65536 Is now set at 4096 in UNIX
MSGMNI 40
MSGTQL 30
MSGSEG 30
MSGSSZ 128
$ exit
```

Operating System Notes and Considerations

Operating System Notes and Considerations

This section contains general observations on specific Unix operating systems that are in use at Inglenet and special operating system considerations that you should keep in mind before you install TIP/ix.

TIP/ix requires the use of the UNIX Streams facilities.

C compiler is required

In order to compile your COBOL applications you will need a COBOL compiler. For online COBOL transaction programs the compile process uses a utility called ‘genmain’ which creates a small C program that acts as an interface between TIP/ix and your COBOL application. The generated C program varies depending on the version of COBOL being used. You will need a C compiler to be installed on the Unix/Linux system.

For Linux there are many optional packages. The packages that you need may vary depending on many factors. A simple solution is to just install them all. TIP/ix does have in the ‘scripts’ directory a script called linuxpkginstall32.sh which uses ‘yum’ to install the needed packages for a 32 bit system and linuxpkginstall64.sh for a 64 bit system. You need to be root in order to run this script and to run ‘yum’. You will also need to enable rlogin and/or telnet on the system in order to use the TIP/fe or TIP/ws terminal emulator.

For AIX, the C compile is an option that needs to be licensed and installed. For versions of TIP/ix prior to “2014/01/17 2.5 R0 – 0262” you need to install LDAP. From version “2014/01/17 2.5 R0 – 0262” on LDAP is only needed to be installed if you plan to use it for TIP/ix to authenticate user logins.

TIP/ix Environment Variables

Shells

The method used to set environment variables depends on which Unix command shell is used.

In the Bourne shell and Korn shell, the environment variable is assigned a value and the **export** command is used to permit the environment variable to be accessed by programs (like TIP/ix) that are run by the command shell.

In the C-shell (csh), the **setenv** command is used to define the value of an environment variable.

Table of Environment Variables

You can also set environment variables in the **tipix.conf** or **.tipixrc** files.

TIPBATCHLOG
: Control logging of BATFCS requests from a batch program.

  a Log all commands, warnings, and details
  c Log commands
  d Log commands
  w Log warnings
  *n*M Recycle the log once it reaches *n* meg.
  For example: export TIPBATCHLOG=a,2M

TIPF01...TIPF22
: Use these environment variables to associate specific character strings to function keys. These settings take precedence over the DEFKEY specification.
  Example: TIPF01="whoson"

TIPFLOW

  OFF By default, flow control is **OFF**. This means that Ctrl-s and Ctrl-q are fed to the TIP/ix shell.
  ON Flow control is enabled. Ctrl-s and Ctrl-q are *not* fed to the TIP/ix shell.
  Note: If you are having problems with data being garbled on terminals or printers (because software flow control is disabled), set TIPFLOW=ON.

TIPDMSLOG
: When this environment variable is defined it will create a log for TIP/dbi. The user is able to set a maximum size for

this log. Variable is defined as follows:
export TIPDMSLOG = k2M # k = keep log.

TIPJRNFLIN
: Define the full name of the journal or QBL file to access with the batch journal file access routines TIPJRNOP, TIPJRNCL, and TIPJRNGT.

TIPLOGDIR
: This variable can specify a directory where the TIP/ix system and program log files will be stored. Setting this environment variable for a given user overrides the system default LOGDIRECTORY. If no log directory is defined, it defaults to the users home directory.
Export TIPLOGDIR = /location/to/put/log files.

TIPMENUKEY
: If this variable is set to "Y", the meaning of Function Key 1 and **MSG WAIT** are reversed for the MENU utility. That is, TIPMENUKEY=Y causes MENU to interpret **F1** as exit from the menu and **MSG WAIT** as refresh the screen.

TIPPATH
: This path specification (if present) is used to define the search path for TIP/ix transactions *and* any Unix programs executed from the TIP/ix command line. Set this environment variable only if the usual PATH variable is not what is wanted.

TIPPRINT
: Defines Unix system printer for use by TIP/ix utilities (like **status**) that may run outside of control of TIP/ix. This environment variable is not used by the TIPPRINT service.

  HP for Hewlett-Packard Laserjet printer.
  EPSON for an EPSON dot-matrix printer.
  D630 for a Diablo 630 printer.
  LP for Unix spooler **lp**.**PS** for Postscript printer.

  You may also specify a postscript filename to send ahead of the print data by specifying **PS=filename** (if no postscript filename is supplied, the default postscript driver named "land" supplied with TIP/ix is used).
  Each of the above values (except PS) may specify any required lp options by appending the options to the printer type.

  Example:
  ```
  TIPPRINT="PS=arcland"
  TIPPRINT="LP -o nobanner"
  ```

TIPPRINTAUX
: Used by the TIP/ix TIPPRINT service or non-TIPPRINT routines to define local terminal printer.

HP
: Hewlett-Packard Laserjet printer.

EPSON
: EPSON dot-matrix printer.

D630
: Diablo 630 printer. Pure nostalgia.

PS
: Postscript printer. You may also specify a filename to send ahead of the print data by specifying **PS=filename** (if no postscript filename is supplied, the default postscript driver -named "land"- supplied with TIP/ix is used).

LP
: (*TIPPRINT service only*) for Unix spooler **lp**. If an application uses TIPPRINT to output to the logical printer named "AUX1" and TIPPRINTAUX=LP is specified, TIPPRINT uses the TIP/ix printer definition for "PRNTR" (as it is defined using SMPRINT).

LNF
: (*for non-TIPPRINT only*). If an application does **not** use TIPPRINT, LNF specifies the name of a print record defined using the SMPRINT utility.

LPP
: (*for non-TIPPRINT only*). This option controls lines per page. Example: TIPPRINTAUX="HP:LPP=45" This example specifies that the AUX1 printer is an HP and sets the lines per page to 45. The LPP option may also be set to "OFF" to disable line counting.
Each of the above values (except PS) may specify **lp** options.

TOF
: If TOF is set, a form feed will be guaranteed at TOP of Form (Before data is printed).

BOF
: If BOF is set, a form feed will be guaranteed at Bottom of Form after data is printed).

TIPPRINTLPP
: Lines per page for system printer. Default value is 64. This environment variable is not used by the TIPPRINT service, but is referenced by programs that run outside of the TIP/ix shell.

Note: There is also a PRINTLPP param variable which may be defined in tipix.conf. The TIPPRINTLPP environment variable overrides the PRINTLPP param. For details, see the chapter *TIP/ix System Configuration File - tipix.conf.*

TIPROOT
: Base directory where TIP/ix is installed.

TIPTERM
: Terminal name to give to application in PIB-TID. This variable is a mechanism whereby a user can force a terminal name that is different than the name assigned dynamically by TIP/ix. This may be necessary to trick older applications that check terminal names.

The following variables are no longer supported:

- TIPJRN
- TIPQBL
- TIPVCDEBUG.

Unix Environment Variables Affecting TIP/ix

PATH
: Should include $TIPROOT/bin in the search path. It may be best to place this setting near the start of the PATH to avoid name conflicts between TIP/ix programs and other programs in your Unix system.

TERM
: Define terminal type to TIP/ix. TIP/ix uses the terminfo version of the Unix curses. If using the TIP/ix Front End terminal emulator (MS-DOS or MS-WINDOWS version), set to **tipfe** (smart mode) or **tipvt** (vt mode). If using another emulator, set to the real terminal type (for example: **vt100**).

LD_LIBRARY_PATH
: See discussion in *TIP/ix Programming Reference*.

LIBPATH (AIX)
: See discussion in *TIP/ix Programming Reference*.

LD_RUN_PATH
: See discussion in *TIP/ix Programming Reference*.

TARGET_BINARY_INTERFACE
: See "Platform-Specific Dependencies" in *TIP/ix Release Notes*.

Micro Focus COBOL Environment Variables Affecting TIP/ix

COBCPY
: Required at *compile* time. Defines search path for copybooks. Should include $TIPROOT/hdr. See *TIP/ix Programming Reference*.

COBDIR
: Required at *run* time. Defines the directory where the COBOL compiler is installed. Usually /usr/lib/cobol. See discussion in *TIP/ix Programming Reference*.

COBSW
: See discussion in *TIP/ix Programming Reference*.

COBOL-IT Environment Variables Affecting TIP/ix

COBOLITDIR
: Required. Defines the directory where the COBOL-IT compiler is installed. Usually /opt/cobol-it.

PATH
: Should include the COBOL-IT bin directory

LD_LIBRARY_PATH
: Should include the COBOL-IT lib directory

Examples of Setting Environment Variables:

For Bourne or Korn shell:

```
COBOL="mf"
TIPROOT="/usr/tipix"
TIPPRINT="PS=land -o nobanner"
TIPPRINTAUX="PS=port"
export COBOL TIPROOT TIPPRINT TIPPRINTAUX
```

For C shell:

```
setenv COBOL "mf"
setenv TIPROOT "/usr/tipix"
setenv TIPPRINT "PS=land -o nobanner"
setenv TIPPRINTAUX "PS=port"
```

For COBOL-IT using C shell:

```
setenv TIPROOT "/usr/tipix"
setenv COBOLITDIR /opt/cobol-it
setenv COB "COBOL-IT"
setenv LD_LIBRARY_PATH ${COBOLITDIR}/lib:${LD_LIBRARY_PATH}
setenv PATH "${COBOLITDIR}/bin:${PATH}"
```

From tipix.conf:

```
COBOL=mf
TIPPRINT=PS=land -o nobanner
TIPPRINTAUX=PS=port
```

TIPROOT must be set before TIP/ix can read *tipix.conf*, so there is no point in setting TIPROOT in it.

Application Compile and Link

TIP/ix includes a couple of utility programs that are used for compiling and linking the COBOL application programs to create exectables.

genapi – is used normally after installing a new release of TIP/ix and this creates and compiles the many 'stub modules' used to interface between the TIP/ix API modules which as all written in C and the customer applications which may be compiled using different COBOL compilers.

genmain – is used to create a small C main program which is compiled and linked with the COBOL application. This 'main' module gets thing connected so that the COBOL program can interface with TIP/ix via the API library.

genmain - Write Unix main Program

The **genmain** utility writes a small C language **main** program that calls the real (COBOL) transaction program. Every program on UNIX must have a function called **main**. All transaction programs have a LINKAGE SECTION and are written to be subroutines of the transaction monitor (TIP/ix in this case).

genmain is normally used in the **makefile** as part of the compilation procedure for online COBOL programs (see references to "makefile".) **genmain** is not normally invoked at the command line!

Usage:

```
TIP/ix 2014/09/11 2.5 R0 - 0274   © 1991-2014 Inglenet Business Solutions

usage:  genmain [-bimsSvxcogG] module mainname.c
Options:
        i   Generate/compile 'main' for IMS/90 (default: TIP/ix)
        1   Generate/compile 'main' for TIP/2200 transaction
        h   Generate/compile 'main' for TIP/HVTIP OS2200 transaction
        c   Generate/compile 'main' for TIP/ix C transaction
        m   Using Micro Focus COBOL
        o   Using OpenCOBOL 1.1 or later, or GNUCobol or COBOL-IT
        X32 Using 32 bit mode
        X64 Using 64 bit mode
        f   Insert 'cobinit' for MF Cobol

        G ssname   Sub-System Entry name
        g nprms    Sub-System LINKAGE parameters used
        s   Create TIPSUBP/SUBPROG from archive
        S   Create TIPSUBP/SUBPROG and update archive
Note: For -n, -s & -S   setenv CCOPTS  to cc options
```

Where:

opts genmain can often guess the transaction type by looking at the object module. However, genmain accepts the following explicit options:

-h An OS2200 TIP/HVTIP transaction program
-i An IMS transaction program.

-t A TIP/ix transaction program.
-c A C language transaction program.
-s A COBOL subroutine.

-n Only for TIP/dbi batch interface to indexed files. Generate a library named libnargs.a.

Since the library is compiler-specific, you should also specify -m or -v. If you don't, the default is Micro Focus COBOL. See the TIP/dbi documentation for details.

-m The Micro Focus COBOL compiler is being used.
-o The OpenCOBOL, GNUCobol or COBOL-IT compiler is being used.

module

The name of the program being compiled. This is the entry point name. genmain will attempt to read an object module of this name and try to determine if it is TIP/ix or IMS, GNUCobol or Micro Focus.

mainname.c

Optional. The name of the "C" source module to be written.

If not supplied, genmain will take the name specified for ***module***. If specified, you must append a ".c" extension.

Example

Generate a C program to call a TIP/ix transaction named myprog (that was compiled with GNUCobol):

```
genmain -to myprog
```

Following is an example of using genmain in a standard 'makefile' for compiling with Micro Focus COBOL.

```
TIPLIBS=-L$(TIPROOT)/$(LIBDIR) -ltip $(MFLIB) -lm
LDBAT=  -L$(TIPROOT)/$(LIBDIR) -lbat $(MFLIB) -ixfile=cixfile +l disam

.cbl:
    cob -c $(FLAGS) -k $(@F).cbl
    genmain -m $(@F) main$(@F).c
    cob $(FLAGS) main$(@F).c $(@F).o -o $(@F) $(TIPLIBS)
    rm -f $(@F).o main$(@F).o main$(@F).c $(@F).int $(@F).idy $(@F).lst
    mv -f $(@F) $(LTIPBIN)
```

Additional Considerations:

For users that used the TIP/30 TIPSUBP utility, which is not supported under TIP/ix you can now take all of your subroutines that were invoked via TIPSUBP,compile them into an archive, then use genmain -s to build a tipsubp.c module that contained names of all the subroutines within the archive.

When you issue a call to TIPSUBP you really enter the tipsubp.c code that grabs the name of the routine you what to call from your PIB, looks it up in its table, and then enters the subroutine.

Generating Sub-System Stub Module

To generate and compile a COBOL routine which is to execute as a standalone process and be called like other subroutines as a running sub-system module you may use the –g and -G options as follows:

-g n Define the number of LINKAGE and CALL parameters that the subroutine uses. If omitted the default is 3.

-G entry[,init,exit]

'entry' is the main entry point name to the subroutine.

'init' is the ENTRY point to be called when the program is first loaded. If omitted it looks for ***entry*-INIT** and if not present there is no init routine called.

'exit' is the ENTRY point to be called just before terminating. If omitted it looks for ***entry*-EXIT** and if not present there is no exit routine called.

Following is an example of using genmain in a standard 'makefile' for compiling with OpenCOBOL.

```
.cblss:
    cobc $(OCFLGS) -c -I$(TIPSRC)/include $(@F).cblss
    genmain -G $(@F) -o $(@F) main$(@F).c
    cobc $(OCFLGS) -x -o $(@F) -I$(ID)/cpy $(TLIB) main$(@F).c $(@F).o
    $(RM) $(@F).o main$(@F).c main$(@F).o $(@F).i
    $(MV) $(@F) $(LTIPBIN)
```

genapi – Generate API stub modules

The genapi utility is used as part of the TIP/ix install procedure. It should also be run whenever a newer version of the COBOL compiler is installed. The modules created depend on the version of COBOL compiler being used. There is one 'stub module' for each application program interface (API) routine used in TIP/ix. The 'stub module' is generated as COBOL code, compiled using the installed COBOL compiler and then object module is then added to the library which the various COBOL applications (batch and online) are then linked with during the compile procedure. Usage:

```
TIP/ix COBOL API generator - © 1991-2014 Inglenet Business Solutions
   for TIPROOT = /u/tipsrc/   [REDHAT6]  Default: 32 bit mode
genapi  Options:
  -l    create lower case API routines as well
  -t    Write log of actions to /tmp/genapi.log
  -O    Generate TIP/ix API library for OpenCobol 1.1 or later
                including GNUCobol and COBOL-IT
  -X32  Operate in 32 bit mode
  -X64  Operate in 64 bit mode
  -Z    Generate Micro Focus Callable Shared Object for TIP/ix 2200 API
 Options for creating Sub-System stub module
  -G ssname   Generate stub module for Sub-System routine 'ssname'
  -g nprms    define how many LINKAGE parameters the 'ssname' routine has
  -L libname  Library that object module is added
```

Usage for install of TIP/ix and recreating the API stub modules:

genapi	with no parameters defaults to generating the API stub modules for Micro Focus COBOL and putting them into the libraries used to compile the COBOL application programs.
genapi -O	with just –O generates the API stub modules for use by the OpenCOBOL, GNUCobol or COBOL-IT compilers.

Generate Sub-System API module

Sub-System modules are subroutines which execute as standalone re-usable processes. The calling programs may call the sub-system module via a generated ‘stub module’. The genapi utility generates the required COBOL code which is then compiled and placed into the declared library. The options used for this follow:

-g n	defines the maximum number of parameters that may be passed on a CALL to this routine.
-G name	defines the name of the subroutine. Calling programs will do CALL “name” USING P1, P2, … Px.
-L library.a	defines the path/name of the library that the generated module is to be added to. Calling programs will then use that library when they are being compiled and linked.

File Recovery

Introduction

In the TIP/ix environment, you can perform a number of file recovery/backup operations to help protect your data.

Offline Recovery

TIP/ix has the ability to record all modifications made to a file in a journal log. If the need should arise, these modifications can be re-applied to a file. The modifications are recorded as AFTER IMAGES in a system journal file.

Typically, data files are periodically backed up to removable storage to protect data, but there is a period of time between data backups where a system failure or corrupted disk might mean the loss of data. However, if the file is managed through the TIP/ix environment and the modifications to the file are being journaled, it may be possible to recover or re-apply the lost changes to the file. This operation is called offline recovery or roll forward.

Online Recovery

TIP/ix allows the grouping of file modifications where a group of changes to a file can be committed or rolled back together. If a file is configured to hold all record locks for the transaction, then as each record is updated a BEFORE IMAGE is stored in the system "quick before look" file (QBL) and the lock on the file record required to make the modification is maintained by TIP/ix. Several modifications can be made and TIP/ix will record each BEFORE IMAGE and hold each record lock until a “transaction end” point is reached. At this point, the data has already been modified, but it is still possible to accept (commit) or undo (roll back) all of the data changes.

To *commit* the updates, TIP/ix releases all of the records locks currently maintained for the transaction. Once this occurs, the changes can’t be reversed.

To *roll back* the modifications, TIP/ix takes the BEFORE IMAGES from the QBL file and re-applies them to the data file, then releases the record locks. Rolling back a group of updates is referred to as online recovery or roll back. See *Transaction End* in the ***TIP/ix Programming Reference***. If a transaction is aborted then any uncommitted updates are automatically rolled back. If the TIP/ix system crashes (possibly due to a Unix system crash or power outage) and there are uncommitted updates, they will be rolled back when TIP/ix is next activated. This is also called "startup recovery".

Journal and QBL file format and usage

The Journal and Quick Before Look (QBL) files are crucial elements in the implementation of file recovery. The journal file contains a record of a sequence of changes made to a file so that the changes can be repeated. The changes can span a long period of time and many transactions. The QBL file contains a record of the original contents of a file’s records so that any changes made to a file by an *uncommitted* transaction may be undone if necessary.

Each record in the journal and QBL file has a header portion and a data portion. The record header contains, among other things, a record type, a time stamp, a transaction identification string and possibly a filename. The record types AFTER and BEFORE are used for modifications to existing records, there are also NEW and DELETE record types for when records are added or removed from a data file.

The Journal file is used primarily to maintain after images of file modifications, before images may be found but they are not functional. There are other types of journal file records, some may be added by user applications to record audit information (via TIPFCS with a function of FCS-JOURNAL), others may be user logon/logoff records (see JRNLGON and JRNLOGOFF parameters), and finally there are

transaction end processing records (not functional). After image records (AFTER, NEW, DELETE) are the only ones used in off-line recovery.

The QBL file is used to maintain relevant before images of file modifications, and transaction end processing records. The transaction end processing records give the system information about the state of a transaction, to indicate if it has reached an end point yet, and if it has, did it commit the updates or roll them back.

For details about journal and QBL file formats, see *Accessing TIP/ix Journal Files* in the ***TIP/ix Programming Reference*.**

Offline Recovery

Offline recovery is useful when you know that a data file has been corrupted, and you have a valid backup of the file. You can restore the backup version of the file and use **rollfwd** to re-apply changes made to the file so that minimal work is lost. See **rollfwd** in the ***TIP/ix Utilities manual.***

The **rollfwd** utility provides *offline recovery* by re-applying the changes that TIP/ix made to a file over a specified period of time. The roll forward utility is driven by the contents of one or more journal files and therefore to be able to perform offline recovery on a file, changes to the file must be journaled.

Enable File Journaling

To enable journaling for a file two conditions must be met:

- The system parameter JouRNal must be set to YES (default). This is a system wide journaling enable command.
- The file must have journaling enabled (see **smfile**). This means that any changes made to the file were recorded in journal file(s). This applies to indexed, direct or sequential files only.

View Journal File

Use **readjrn** to view the contents of the journal file. You might want to direct the output to the **more** or **pg** utilities or to a file.

See the TIP/ix Utilities manual.

Maintain Journal File

To work, the **rollfwd** utility must have journal file information *from* the time of the last valid backup *to* the time of the last valid update to re-apply.

In any TIP/ix system there is one active journal file for the system containing all changes made to any file with journaling enabled. This file can easily grow too large using up disk space and slowing down the system. To prevent this, the journal file itself must be periodically backed up, possibly more frequently than the data files. Backup journal files may

be stored in compressed format, then uncompressed if a file recovery is necessary.

The saving, storing and removing of journal files must be carefully coordinated with data backups. The minimal rule is to have at hand journal file(s) that span the time period from the last (few) successful data backups to the present.

A journal file can't be successfully backed up if it is being used by TIP/ix. To ease and help automate the process of backing up journal files **dual journaling** has been implemented. The concept is to allow the user to define two journal files, even though only one journal file is **active** at a time. To back up a journal file without shutting down TIP/ix the system administrator can swap journal files, activating an alternate journal file and deactivating the current journal file and then backup and empty the previously active journal file. If dual journaling is employed the entire process of swapping and backing up journal files can be automated, see below.

If your site does not have dual journaling enabled, you must shutdown TIP/ix to do a journal file backup.

Single Journal Maintenance:

- Shutdown TIP/ix.
- Backup the journal file.
- Empty the journal file.

To avoid having to shutdown TIP/ix for journal file maintenance, we recommend dual journaling. We also recommend that after backing up the journal files delete the old journal files and start new files. This should be done to conserve space on the machines hard drive.

Dual Journal Maintenance:

- If dual journaling is configured, you have two choices:
- Swap journal files with jrnswap when the current journal file is full, or
- use the JRNSIZE parameter (in tipix.conf) to automate journal file swapping.
- Back up the swapped out journal file.
- Empty the swapped out journal file.
- The backup and emptying of the old journal file can be done with a shell script. The JRNSWAPSCRIPT param can be set up to point to this script and it will be automatically invoked by the jrnswap utility or the automatic swap.

The journal files must be kept in the order of their occurrence. If you need three separate journal files to recover your data you must rollfwd supplying the name of the oldest journal file first, then rollfwd using the name of the second oldest journal file, and finally rollfwd with the newest journal file. The rollfwd utility accepts as an argument the name of the journal file to process to complete it's operation, if no argument is

supplied it will use the journal files specified in the **tipix.conf** file or the default journal file name.

In a very busy system it may be necessary to backup journal files frequently. In this case, you may want to concatenate the backup journal files together into a single file to simplify storage. Be sure that as the each journal file is backed up, it is **appended** to the backup copy of the journal file.

For a journal file to be used successfully for recovery, the changes must appear in the proper real time order, **readjrn** can be used to verify this.

Be sure to test any journal file backup scripts thoroughly, your data recovery depends on it.

Set Journaling Parameters

There are number of system parameters (JRN...) in the **tipix.conf** file that can be used to help in the maintenance of journal files. They allow the system administrator to specify the names (and locations) of journal files(s), the enabling of dual journaling, the automation of journal file swapping and journal file backups.

See TIP/ix System Configuration File section in the **TIP/ix Installation and Operation manual**.

Steps to perform offline recovery

To recover a corrupted file:

- Take the file(s) offline. Either: shut TIP/ix down with **tipctl**, or **fclose** the files.
- Restore file(s) from backup copies.
- Determine and retrieve the journal files required. Use **readjrn** to display date and time values.
- Apply the updates from the journal file(s) with **rollfwd**.
- Put file(s) back online. Either: startup TIP/ix with **tipctl**, or **fopen** the files.

Online Recovery

Online Recovery or roll back can be:

- requested explicitly (manually) by a transaction via the TIPFCS FCS-TREN function call.
- invoked automatically for aborted transactions
- invoked automatically on TIP/ix startup after a system failure to cleanup any uncommitted transactions.

Roll back is completed by changing data records back to their original contents, or removing them if they did not previously exist.

Online recovery is driven by the contents of one or more QBL files. Thus, to be able to perform online recovery on a file, *before images* of modified records in the file must be recorded in the QBL files. Online recovery is also transaction based and once a transaction has completed its modifications cannot be undone, for this reason records are added to the QBL file to indicate when a transaction has completed.

Enable File Recovery

To enable roll back for a file:

- The file must have the record hold attribute set to T (Hold for Transaction, see **smfile**). This means that any updates to the file are recorded with before images in the qbl file and that record locks are maintained for the duration of the transaction. This applies to indexed, and direct files only.

View QBL Files

Use **readjrn -q** to view the contents of the QBL files. You might want to direct the output to the **more** or **pg** utilities or to a file.

See the *TIP/ix Utilities* manual.

Maintain QBL Files

Unlike the journal files which grow infinitely the QBL files are self-compressing. At any point in time the only information that is important in a QBL file are records of updates for uncommitted transactions. Any before images for committed transactions are of no use since they can't be rolled back. The QBL file can and does periodically compress itself by removing any unnecessary records. If there are no transactions outstanding in the system and they have gotten large enough the QBL files will be emptied, otherwise system parameters can be used to control the frequency of QBL file packing or compression.

To increase performance several QBL files (up to six) may be used by TIP/ix, each file could be placed on a different device to decrease file access times.

Set QBL file Parameters

There are number of system parameters (QBL...) in the **tipix.conf** file that can be used to help in the maintenance of QBL files. They allow the system administrator to specify the names (and locations) of QBL files(s), an interval for checking the size of QBL files, and the maximum size for QBL files.

See TIP/ix System Configuration File chapter in the **TIP/ix Installation and Operation manual**.

Steps to perform online recovery

Automatic online recovery:

- If a transaction aborts any uncommitted updates are automatically rolled back.
- If the TIP/ix system crashes unexpectedly any uncommitted updates are automatically rolled back at the next startup of TIP/ix.

Manual online recovery:

- A transaction may request that all updates are rolled back via the TIPFCS FCS-TREN function (by setting a flag in its PIB).

See the **TIP/ix Programming Reference** manual sections PCS: Transaction End, TIPFCS: FCS Miscellaneous Functions (FCS-TREN).

Recover Corrupted Journal/QBL files

Beginning with release 2.0, it is possible to have several qbl files spread out over several disks to enhance system performance. It is also possible to automatically limit the size of the active journal file, using a journal swap and backup method, to enhance system performance and preserve disk space. However, even with these extra features it is possible for TIP/ix to find itself in a situation where it is unable to write to a QBL or journal file, at this point data integrity is at risk.

With release 2.3 the system will shut down immediately if an error is detected handling a journal or QBL file. As well, after an abnormal system shutdown all of the journal and qbl files can be scanned with the readjrn utility to ensure that they have not been damaged and if they have been, the utility may be able to correct the file.

A possible scenario is running out of space on the disk containing a journal or qbl file. Once an error is detected writing to a journal or qbl file, the system is immediately shutdown, and any other journal or qbl file writes are prevented. This is the safest and really the only way to handle this situation. In previous releases TIP/ix would shutvdown but the action was not immediate and controlled.

If the system has shut down due to a QBL or journal file handling error or there is a Unix crash, the file may be corrupted, very often this corruption is simply a trailing partial record, but this does make the file in question invalid. If this happens to a qbl file and a startup is attempted the startup recovery will fail, because it is not safe at this point for recovery to assume the nature of the problem in the QBL file and allow startup to continue.

To help determine the nature of the problem the **readjrn** utility (which is used to scan journal and qbl files) has been enhanced to detect file corruption. It attempts to dump the corrupted data so that you can determine if the problem is simply a trailing partial record or something more serious. Optionally, **readjrn** can attempt to correct the file. In general, **readjrn** will delete the bytes following the detected point of corruption. This enables the system to startup and recover successfully.

For more information about journal and QBL file correction, see the **readjrn** utility in the ***TIP/ix Utilities manual***.

Trouble Shooting Guide

Reproducing MCS issues

When trying to reproduce an MCS issue, it is important to be aware that the version of TIP/ix being used is only one factor. The MCS code resides in both TIP/ix and the emulator (TIP/fe or TIP/ws) when running in "smart mode"

In order to reproduce a problem, first try to isolate it on just the TIP/ix side. You need to start your TIP/ix session with tipixtty . With tipixtty you are in ASYNC mode and the MCS code is being executed on TIP/ix only.

If the problem can't be reproduced with tipixtty but you can with tipix (which calls tipixtcp) then you have isolated the problem to be with the emulator, TIP/fe or TIP/ws.

To reproduce a MCS coding problem in TIP/fe the tipixtcp module must be executed.

Extensive Load on CPU

Sometimes a user can take upwards of 90% of a CPU. If you find this happening on a regular basis follow the following steps to pinpoint what exactly the problem is:

- First see what file is getting a lot of hits on the server.
- To find the server, do a pingtip to see what server had the process id # that you are looking for (use the UNIX command 'top' (on AIX topas)). You should see a -1 or -2 or some number at the beginning of the line. Use that number and do a status -s# which should show the files under the server.
- Then you can repeat the status a few times to see if a file I/O count is increasing dramatically.
- Then look in the tipfcs log to see which user might be causing the majority of the I/O.
- When the user ID is found issue the 'whoson userid' to determine the programs they are using.

Check The Log Files

TIP/ix has many options to record a log of the system activity. The TIP/ix internal system log files are in the $TIPROOT/log directory. You will also find a file called HISTORY in $TIPROOT/log which contains information about who/when the system was started/stopped. Also some critical information gets recorded to this HISTORY file.
PRINTER is a file which holds any messages created by the transaction application programs via DISPLAY UPON PRINTER and/or DISPLAY UPON CONSOLE. This can also be a technique for a programmer to debug their program by inserted DISPLAY UNPON PRINTER statements and then removing them after the logic error has been found.
Some application log files may also be created in the $TIPROOT/tmpwrk directory. If you need this information grab it when you can as the ‘tmpwrk’ directory does get emptied each time the TIP/ix system is started up.
If you have TIP/dbi logging turned on via the TIPDMSLOG environment parameter in tipix.conf then all TIP/dbi activity is logged into the $TIPROOT/log directory into files called dbi.schema.pid.

COBOL Sub-System management

A ‘Sub-System” (is patterned after a feature of OS2200) is a method of having a COBOL subroutine execute as an independent process and maintain its state information from call to call. Such a module may optionally have an initialization routine called on initial start-up and an exit routine called just before the module is shutdown.

Standard Unix/Linux shared code have a single copy of the module instructions but each caller gets their own local copy of the data used by a ‘shared code’ subroutine.

A ‘sub-system module’ is a separate process and is executed on behalf of one caller at a time. There is only one copy of the subroutines data which is used from one call to the next. This allows a ‘sub-system module’ to retain and reuse data which is shared by all callers.

There is a utility called ‘**ssctl**’ which may be used to start or stop the whole system, start or stop individual modules and also display status information.

The program ‘**ssmonitor**’ runs as a daemon and is responsible for monitoring all programs which are running as ‘sub-system’ routines.

A sub-system module is a standalone COBOL subroutine which runs as a single process on the Unix/Linux operating system. Such a program is executed on behalf of one CALLer at a time, while the state information of the subroutine is retained from call to call. So all of WORKING STORAGE, all files (FD) and any database connections are kept open from one call to another. Such routines may be used to keep tables of information for quick ‘in memory’ access or to have a complicated database process or whatever you can dream up.

Sub-System modules run independent of TIP/ix so these may be used regardless of whether TIP/ix is running or not. The calling program may be a batch program or an online transaction program. The system is managed using shared memory, message queues and semaphores. If all of the data from all CALL parameters can be packing into a single message then it is passed thru the Unix/Linux message queue. If it will not fit, then the data of the CALL parameters is copied to shared memory and passed that way. Each sub-system module has a single message queue assigned to it.

When **ssmonitor** starts up it reads configuration information held in the file called /etc/subsystem.conf. An example of this file follows:

```
# Inglenet Sub-System control file
LOG=Y
LOGOPTS=ath
SETENV FOO Bar

# Define each module to be managed
[test1]
SETENV DD_TSPFILE ${TIPROOT}/tipfiles/tspfile
PATH=/usr/local/bin:/u/tipsrc/bin64:/u/tipsrc/bin
LOAD=tstss1
LOG=Y
RESTART=YES
LINKAGE=500
ENTRY=(TEST1Y,TEST1X)
USER=oracle

[test2]
SETENV COW Dune${TIPROOT}
unsetenv HooHaw
@asg my*file
PATH=/usr/local/bin:/u/tipsrc/bin64:/u/tipsrc/bin
LOG=/tmp/ss2.log
LOGOPTS=aht7M
LOAD=tstss2
```

The parameters at the start of the file define general options for entire system. Each module is defined with its name in square brackets and followed by options which apply to that module. A module may also be qualified with a group name and when the environment variable SSGROUP is defined any CALLs to sub-system modules will result in called the given module name which is also defined with the declared group name.

SETENV may be used in the general section to define environment variables set for all modules or when defined in a specific module section then the variable is only defined in the environment of that module.

UNSETENV may be used to remove environment variables for all or just for specific modules.

@eclcommand may be used to manage various ECL options for the module to be executed. Such as @ASG, @DELETE, etc and these are executed just before loading the module.

At run-time, all UNSETENV are done first, then all SETENV and then all @ecl statements.

General parameters may be any of the following:

GROUP=name	Define the default module group name to be used.
LOG=file/path	Create a log file which the given name
LOG=Y	Create log file /tmp/ssmon.log for ssmonitor
LOGOPTS=options	Define logging options to be used. Logprint standard options apply.

Specific module parameters may be any of the following:

ENTRY=othername	Name (or list of names) of alternate calls names this same module may be called by.
GROUP=name	Define the module group name to be used. If omitted the general group name is used.
LINKAGE=num	Defined total size of all parameters passed on a CALL. If this number is low, it will be adjusted up as required.
LOAD=name	Name of the executable to load. This would have been compiled using the genmain –G options.
LOG=file/path	Create a log file which the given name
LOG=Y	Create log file /tmp/*loadname*.log
LOGOPTS=options	Define logging options to be used. Logprint standard options apply.
PATH=path:list	Colon separated list of paths to be used for searching for the module to load
RESTART=Y|N	Yes indicates to restart the module should it ever terminate No indicates to not restart the module if it terminates
USER=userid	'userid' is the Unix/Linux user that the module is to execute as. 'ssmonitor' must be setuid root for this feature to work.

genmain is used to create the 'main' C module which sets things up for the module to run as a 'sub-system'. See genmain on page 82 for details.

genapi is used to create the API stub module that calling program are linked with in order to call the module as a sub-system. See genapi on page 84 for details.

ssctl usage

Once you have created the /etc/subsystem/conf file and written, compiled and linked your subsystem modules, then next step is to start the system up. The ssctl utility has several commands.

```
TIP/ix ver 2014/09/29 2.5 R0-0276 © 1991-2014 Inglenet Business Solutions
    SubSystem control program
ssctl [options]
 -b        Boot (startup) sub-system monitor
 -s        Stop the entire sub-system & monitor
 -S        Forced Stop of the entire sub-system & monitor
 -v        Display Sub-System details
 -t name   Stop the 'name' module only
 -r name   Stop the 'name' module and reload it
 -e name   Enable the 'name' for use
 -d name   Disable the 'name' module from use
```

To view the current status use the ssctl –v command:

```
TIP/ix ver 2014/09/29 2.5 R0 - 0276  © 1991-2014 Inglenet Business
Solutions
    SubSystem control program
SubSystem IPC Key: 41012F1A, Main queue: 14778368  Monitor Pid: 18610
Name           Pid       Qid    #calls  Msgsz --Flags--
TEST1        18612  14811137         8    500 restart
    Extra ENTRY TEST1Y, TEST1X
QA/TEST1     18613  14843906         0    500 restart

TEST2        18614  14876675         3    104 restart
```

To stop the system use ssctl –s:

```
    SubSystem control program
    Sending Stop request to TEST1  Q14811137:P18612
    Sending Stop request to QA/TEST1  Q14843906:P18613
    Sending Stop request to TEST2  Q14876675:P18614
Sub-System monitor asked to shutdown
```

To start the system back up use sstcl –b:

To stop one single module:

```
ssctl -t QA/TEST1

    SubSystem control program
    Sending Stop request to QA/TEST1
```

To restart the module:

```
ssctl -r QA/TEST1

    SubSystem control program
Requested restart of QA/TEST1
```

If the module is running and you want a new copy reloaded:

```
ssctl -r QA/TEST1

    SubSystem control program
Requested reload of QA/TEST1
```

Connection Server

The 'connection server' consists of two programs:

csctl	is used to start, stop, display status and changes to the system
csmon	is the primary runtime module. This will handle many different TCP/IP sockets, redirect data from one place to another and optional record/playback messages.

When starting up the connection server a configuration file is read by **csmon** and that directs its operation. For TIP/ix using Conversational SOAP, TIP/ix will create a starter csmon.conf file and start **csmon** automatically. If you want more control, then you can make up your own **csmon** configuration file and add to tipix.conf PARAM CSMON=MANUAL. The default configuration file created by TIP/ix will be called csmon.conf and stored in $TIPROOT/conf directory.

Configuration Parameters

The configuration file is divided into sections. Comment lines start with a #. The first group of parameters define options global to the operation of the connection server. Then you define individual connections and then define paths from connection to connection. For example:

```
# Connection Server Config file
LOG=Y
LOGOPTS=ht5M
LOGDIR=/u/tipsrc//log

[*Connections*]

[browser]
HOST=localhost
LISTEN=8082
PROTOCOL=SOAP
METHOD=TRANSIENT
SESSION=MySession_Id

[tipix]
CONNECT=localhost:18082
PROTOCOL=SOAP
METHOD=PERSISTENT

[*Path*]

[soapthru]
PRIMARY=browser
ACTIVE=tipix
```

The **[*Connections*]** marks the end of global options and start of connections. The **[*Path*]** marks the end of connections and start of pathways.

Global parameters

ADDCLIENT	N	If you do not want the client IP address inserted to SOAP HTTP messages. Default: Y
BUFSIZE	nnn	Initial size of I/O buffer allocated. Default: 2560
CONSOLE	filepath or device	where to send console messages
HISTORY	filepath	where to record history events. Default is $TIPROOT/log/HISTORY
LOG	Y	if you want a log file created for debugging
LOGDIR	directory	declare where to create the log file(s)
LOGMSG	Y	If you want the data for each message logged. Default: N
LOGOPTS	opts	declare any of the standard TIP/ix logging options
MAXSERVER	n	Due to limits on open files per process, csmon will spawn additional processes as needed. Defines a limit on how many to start. Default: 64
MAXSESSION	n	Define maximum number of concurrent sessions. Default: 2048
SSLCERT	filepath	Define complete path to a valid SSL certificate file. Default: $TIPROOT/tipsoa/certs/cacert.pem
SSLCERTDIR	directory	Define location where SSL certificates are kept. Default: $TIPROOT/tipsoa/certs
SSLKEY	filepath	Define complete path to valid SSL Key file. Default: $TIPROOT/tipsoa/certs/privkey.pem

For SSL, you need to have an SSL certificate. This could be created using openssl or any other SSL certificate authority. For example:

```
    cd $TIPROOT/tipsoa/certs
    openssl genrsa > privkey.pem
Then if using a certificate authority:
    openssl req -new -key privkey.pem -out cert.csr
and send the request file to get authorized
If using a local test certificate:
    openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1500
```

Connection definition

Each connection starts with [name] where 'name' is the name of the connection. This is then followed by parameters that define that connection.

CONNECT	port	defines the TCP/IP port to connect to
DATA	type	Define default data type: JSON or XML
DISABLE	Y	Initially disable the connection
HOSTNAME	Host/IP	TCP/IP network name or IP address
LISTEN	port	defines the TCP/IP port to listen on
LOCATION	filepath	defines location of data
METHOD	TRANSIENT	Connection is transient as how a Web Browser operates
	PERSISTENT	Connection is persistent as with TIP/ix user interface
PASSWORD	pwd	the 'password' used to encrypt the data
SESSIONID	tag	'tag' is used to identify a 'session id' in the message
PROTOCOL	FILE	This defines a data file used for recording messages
	HTTP	Defines a HTTP protocol is used
	HTTPS	Defines HTTPS protocol using SSL
	SOAP	Defines SOAP/HTTP protocol
	SOAPS	Define SOAP/HTTPS protocol using SSL
ALLOW	IP address	IP address (pattern) which is allowed to connect
DENY	IP address	IP address (pattern) which is denied connection

An 'IP Address pattern' may include an asterisk which represents a wild card and will match any number of digits in the IP address being tested.

Path definition

PRIMARY	Connection	Name the primary connection which is handled as the input side
SECONDARY	Connection name	Name the secondary connection which has data passed to as output Responses from 'secondary' are forwarded back to the 'primary'
PASSIVE	Connection names	Other connections which may be activated as required.
RECORD	Connection	Which is used to record messages
COMPARE	Pattern	Define patterns used for comparing responses and allow differences. # means any digit is allowed. . means any character is allowed.

An example of more complex config file follows:

```
MAXSESSIONS=512
[*Connections*]
[saveit]
PROTOCOL=FILE
LOCATION=/tmp/httpdata
[SoapLogs]
PROTOCOL=FILE
LOCATION=/tmp/savedsoap
[browser]
LISTEN=8085
PROTOCOL=SOAPS
METHOD=TRANSIENT
SESSION=MySession_Id
ALLOW=192.168.*,192.168.1.*
ALLOW=192.168.100.*
DENY=172.*
[tipix]
HOST=localhost:8082
PROTOCOL=SOAP
METHOD=PERSISTENT

[centaur]
HOST=centaur:18082
PROTOCOL=SOAP
MODE=STANDBY
METHOD=PERSISTENT

[*Path*]
[soapthru]
PRIMARY=browser
SECONDARY=tipix
PASSIVE=centaur,popa
RECORD=saveit
COMPARE='UPDTM/#####'
COMPARE='ENDTM/#####'
COMPARE='##*'
COMPARE="FRED\# Flintstone\\"
```

csctl Operation

The ‘csctl’ program is used to manage operation of the connection server (csmon).

To start the connection server using a configuration file:

```
csctl boot $TIPROOT/conf/csmon.conf
```

To start the connection server using the default configuration file /etc/connserv.conf:

```
csctl boot
```

To stop the connection server:

```
csctl stop
```

To display status of running connection server:

```
csctl s
```

Example response:

```
TIP/ix ver 2014/12/10 2.5 R0 - 0287  © 1991-2014 Inglenet Business Solutions
    Connection Server control program
IPC Key: 0x43019173  Monitor Pid: 4889  Version: 4.32 2014/12/14
Semaphore: 23625732  Sessions: 0
Now is 2014/12/17 11:55    Server Started 2014/12/17 11:55
Name       Proto        Action  Host
=========  ======    =========  ==================
BROWSER    SOAPS        LISTEN  Port 8085                  SessionId Transient Active
TIPIX      SOAP        CONNECT  localhost:8082             SessionId Persistent Active
CENTAUR    SOAP        CONNECT  centaur:18082 192.168.1.32 SessionId Persistent Standby
POPA       SOAP        CONNECT  popa:8082 192.168.1.22     SessionId Persistent Standby

Name       File name
=========  ========================
SAVEIT     /tmp/httpdata
SOAPLOGS   /tmp/savedsoap

Name       Input      Active     Passive(s)
=========  =========  =========  ================
SOAPTHRU   BROWSER    TIPIX      CENTAUR    POPA       Record:SAVEIT

Server P4889 has 1 open sockets & 0 active sessions
```

If a path was defined with a ‘recording file’ but it is initially turned off, you can start the recording with the following command.

```
csctl record [path [TO connection-file]]
```

If ‘path’ is omitted and there is only one path defined that is used. The ‘connection-file’ is an already defined connection of type FILE. This is enabled and file opened and then setup as the recording file for the path. Example:

```
csctl record soapthru to soaplogs
```

If you want to disable a specific connection:

```
csctl disable connection-name
```

If you want to enable a specific connection:

```
csctl enable connection-name
```

When you want to bring the passive system online and start replaying recorded transaction messages into the passive system:

```
csctl online [path [connection-name]]
```

If path is omitted the only path defined is assumed. If connection-name is omitted then the first passive system defined in the path is assumed.

If you need to take the passive system offline, stop playback but keep recording:

```
csctl offline [path [connection-name]]
```

If you want to toggle the active and passive systems:

```
csctl toggle [path]
```

chgcode – DMS database and program changes

'chgcode' is a TIP/ix utility program which is able to somewhat automate changes to a DMS database (as implemented with TIP/dbi II on Unix/Linux using Oracle) as well as the COBOL application code changes and any DPS screen changes. The program usage is as follows:

```
Code change impact analysis - Version 4.6 2012/11/06
      © 1991-2012 Inglenet Business Solutions
chgcode [options] -i commandfile
  -d    Do make code changes; Default is no actual change
  -c    Compile updated programs
  -f    Change FLDP-nnn instead of DPS00nnn.t3b
  -F    Change DPS00nnn.t3b instead of FLDP-nnn [This is the default]
```

The **-d** option would result in the program code to be actually changed. The DMS schema is changed and all subschemas recompiled, program source is changed and DPS screens are changed by updating the FLDP text and then rerunning 'dpscnv' to recreate the screen format and DPS COPY book. The default is that the possible changes are searched for and recorded in the report file.

The **-c** option will make all code changes, update the Oracle database table definitions and then recompile each program that was changed.

The **-F** causes screens to be changed by updating the DPS00nnn.t3b file instead of the FLDP-nnn file. If you have changed screens using the Windows TIP TFD program, this updates the DPS00nnn.t3b file but not the FLDP-nnn file.

When you ask for the code to be changed and you are using CVS to manage the code changes, then if the file is 'read only' and there is a CVS subdirectory, then 'chgcode' will invoked the 'editmod' script to check the module out of CVS.

A sample 'command file' follows:

```
TIPSITE = "/projects/nebf"

SCHEMA  nebf
RECORD  NECA-REGION
FIELD   NECA-ADDR1  PICTURE X(30)
FIELD   NECA-ADDR2  PICTURE X(30)

RECORD  EBB-DATA
FIELD   EBB-ADDR1   PIC X(30)
FIELD   EBB-ADDR2   PIC X(30)
FIELD   EBB-CITY    PIC X(20)

COPY  FD-PAYROLL
FIELD   EMP-ADDR1   PICTURE X(30)
FIELD   EMP-ADDR2   PIC  X(30)
```

The command file has several optional keyword parameters. If TIPSITE, TIPDMS, TIPROOT is not defined in the command file the values will be taken from environment variables.

The possible keyword parameters are as follows:

TIPSITE	Define base directory of where the application code is stored
TIPROOT	Define Tip/ix base/install directory location
TIPDMS	Define location of TIP/dbi II data dictionary directory
SOURCEDIR	Colon separated list of subdirectories to be searched for the COBOL source code. Default is "online:batch:sub"
COBEXT	Colon separated list of program file extensions. This is used to determine which modules to look at. Default is "dml:dmlsql:dmo:dmosql:dmou:dmousql"
SCHEMA	Defines the DMS schema name to receive the data format changes
RECORD	Name of a record in the database to be changed
FIELD	The field name to be changed
PICTURE	The new PICTURE clause for the data field
COPY	The name of a COPY book which has fields to be changed. The following FIELD & PICTURE parameters apply to this COPY book.

Appendix A - Micro Focus File Locking

LOCK MODE IS EXCLUSIVE = lock on whole file
LOCK MODE IS AUTOMATIC = lock on single record
LOCK MODE IS MANUAL = lock on single record
Default when No compiler directive specified and No LOCK MODE statement = Lock on whole file when opened for I-O or OUTPUT
This chart is the same for both Relative and Indexed Files.

Batch Job B

Batch Job A		None			Exclusive			Shared		
		Input	I-O	Output	Input	I-O	Output	Input	I-O	Output
None	Input	**Read**	Wait file	Wait file	Wait file	Wait file	Wait file	**read**	**Read Write**	**Read Write**
	I-O	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file
	Output	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file
Exclusive	Input	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file
	I-O	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file
	Output	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file	Wait file
Shared	Input	**read**	Wait file	Wait file	Wait file	Wait file	Wait file	**read**	Wait rec	Wait rec
	I-O	**Read**	Wait file	Wait file	Wait file	Wait file	Wait file	Wait rec	Wait rec	Wait rec
	Output	**Read**	Wait file	Wait file	Wait file	Wait file	Wait file	Wait rec	Wait rec	Wait rec

TIP/ix can be configured to open files as Exclusive, Shared or No file locking through the SMFILE utility.

Appendix B - Sperry UTS FCC codes

Primary FCC processing (UTS400, UTS20):

Name Bit(s) Description

fccM 1,0 Intensity 00=Normal
01=Off,
10=Low/Reverse (by Ctrl page option)
11=Blinking
2 Changed 0=changed, 1=not changed
3 Tab stop 0=tab, 1=no tab
4 Reserved always 1
5 Reserved always 1
6 Reserved always 0
7 Reserved always 0

fccN 1,0 Data entry type:00 Unrestricted entry
01 Alphabetic only
10 Numeric only
11 Protected
2 Right-just:0=none, 1=right just. bit ignored if field is protected
applies to numeric as well as alpha fields.
3 Reserved always 0
4 Reserved always 1
5 Reserved always 1
6 Reserved always 0
7 Reserved always 0

Expanded FCC processing (UTS40 & UTS60)

fccM 0 Video 0=On, 1=Off
1 Intensity 0=Normal, 1=Low
2 Changed 0=Changed, 1=Not changed
3 Tab Stop 0=Tab, 1=No tab
4 Kanji 0=Alphanumeric/Katakana, 1=Kanji
5 1=Special emphasis not-protected, 0=Special emphasis protected
6 Reserved always 1
7 Reserved always 0

fccN 1,0 Data entry type:00 Unrestricted entry
01 Alphabetic only
10 Numeric only
11 Protected
2 Right-just:0=none, 1=right just. bit ignored if field is protected
applies to numeric as well as alpha fields.
3 Blink 0=No Blink, 1=Blink
4 Intensity 0=Normal, 1=Reverse
5 Kanji 0=Not Kanji input only, 1=Kanji input only
6 Reserved always 1
7 Reserved always 0

UTS 60 Colour FCC

These are 7 bytes long. 0x1F rr cc mm fccM fccN clr

Where **rr** is the row, **cc** is the column,

and **mm** is always a 0x20 to indicate 7 byte FCC

Name Bit(s) Description

'fccM' is basically the same as fccM above for UTS40

- 0 1 = Video Off, 0 = Normal
- 1 1 = Low intensity, 0 = Normal
- 2 Changed 0=Changed, 1=Not changed
- 3 0=Tab stop, 1=No Tab Stop
- 4 Reserved: always 0
- 5 1=Special emphasis not-protected, 0=Special emphasis protected
- 6 Reserved: always 1
- 7 Reserved: always 0

'fccN' is basically the same as fccN above for UTS40

- 1,0 Data entry type:00 Unrestricted entry
  - 01 Alphabetic only
  - 10 Numeric only
  - 11 Protected
- 2 Right-just:0=none, 1=right just. bit ignored if field is protected
  - applies to numeric as well as alpha fields.
- 3 Blink 0=No Blink, 1=Blink
- 4 Intensity 0=Normal, 1=Reverse
- 5 Kanji 0=Not Kanji input only, 1=Kanji input only
- 6 Reserved always 1
- 7 Reserved always 0

'clr' is the color values

- 2,1,0 Foreground Color
  - value 0 – Ebony, 1 – Red, 2 – Green
  - 3 – Yellow, 4 – Blue, 5 – Magenta
  - 6 – Cyan, 7 - White
- 5,4,3 Background color
- 6 Reserved always 1
- 7 Reserved always 0