HP NonStop AutoTMF Software User's Guide

Abstract

HP NonStop™ AutoTMF™ Software enables programs that are not configured to use the HP NonStop™ Transaction Management Facility (TMF) product to access and update audited databases. This manual describes the uses, installation, and all other operational aspects of NonStop AutoTMF Software.

Product Version

NonStop AutoTMF Software AAS

Supported Release Version Updates (RVUs)

This manual supports all J-series RVUs, all H-series RVUs, and all G-series RVUs, until otherwise indicated in a replacement publication.

Part Number	Published
429952-014	June 2011

Document History

Part Number	Product Version	Published
429952-010	NonStop AutoTMF AAO (Update 8)	April 2008
429952-011	NonStop AutoTMF AAP (Update 9)	April 2009
429952-012	NonStop AutoTMF AAQ (Update 10)	January 2010
429952-013	NonStop AutoTMF AAR (Update 11)	October 2010
429952-014	NonStop AutoTMF AAS (Update 12)	June 2011

Legal Notices

Printed in the US

HP NonStop AutoTMF Software User's Guide

Glossary | Index | Tables

6. NonStop AutoTMF Software Commands

D. Error Messages

What's New in This Manual

Manual Information

Abstract

HP NonStop™ AutoTMF™ Software enables programs that are not configured to use the HP NonStop™ Transaction Management Facility (TMF) product to access and update audited databases. This manual describes the uses, installation, and all other operational aspects of NonStop AutoTMF Software.

Product Version

NonStop AutoTMF Software AAS

Supported Release Version Updates (RVUs)

This manual supports all J-series RVUs, all H-series RVUs, and all G-series RVUs, until otherwise indicated in a replacement publication.

Part Number	Published
429952-014	June 2011

Document History

Part Number	Product Version	Published
429952-010	NonStop AutoTMF AAO (Update 8)	April 2008
429952-011	NonStop AutoTMF AAP (Update 9)	April 2009
429952-012	NonStop AutoTMF AAQ (Update 10)	January 2010
429952-013	NonStop AutoTMF AAR (Update 11)	October 2010
429952-014	NonStop AutoTMF AAS (Update 12)	June 2011

New and Changed Information

- The UPDDLLS TACL macro has been added to the AutoTMF operational subvolume. UPDDLLS updates the rebased and prepared public host language runtime DLLs in the product subvolume. For details, see Upgrading the Operating System on page 2-10 and New Versions of DLLs and Operating System Upgrades on page A-15 under TNS/E Host Language Runtime DLLs.
- A new global parameter, MAX[MON]OPENS has been added to configure the maximum number of simultaneous OPEN requests the monitor can process. See ALTER GLOBAL on page 6-34 for a description of the new option.
- The description of the COPY command options has enhanced. See COPY on page 6-46.

- Vaious minor corrections have been added throughout the document, based on reader input.

About This Manual

This manual describes the uses, installation, and all other operational aspects of the HP NonStop™ AutoTMF™ software.

This software product enables programs that are not written to use the HP NonStop™ Transaction Management Facility (TMF) product to access and update audited databases. It automatically manages TMF transactions required to access audited data and enables a rapid migration to an audited database.

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

To preserve the modification timestamp, see Preserving the Modification Timestamp of Object Files on page 3-10.

General Syntax Notation

The following list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on

each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num ]
   [ -num]
   [ text]

K [ X | D ] address-1
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address-1 [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must enter as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In the following example, there are no spaces permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

Change Bar Notation

Change bars are used to indicate substantive differences between this edition of the manual and the preceding edition. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1

Introduction to HP NonStop AutoTMF Software

This section of the manual discusses the following topics:

- An overview of HP NonStop AutoTMF software
- A description of the facilities that are enabled by NonStop AutoTMF software, both immediately and through a migration process

HP NonStop AutoTMF software enables programs that are not programmed to use the HP NonStop Transaction Management Facility (TMF) product to access and update audited data. It automatically manages TMF transactions required to access audited data and enables a rapid migration to an audited database.

You must prepare your programs to use NonStop AutoTMF software by invoking a utility that processes object files. This utility inserts a software layer, called the NonStop AutoTMF runtime library, between each program and the file system. This layer automatically manages the transactions required to access audited data.

The preparation utility is automatic and processes collections of object files quickly and economically. Preparation does not require program source and does not change the logic flow of the program.

The migration to an audited database is incremental. You audit your database files as desired; NonStop AutoTMF software determines which files are audited and makes the necessary transaction operations whenever necessary.

In providing a migration path for applications that do not currently use audited data, NonStop AutoTMF software offers several useful features:

- Because NonStop AutoTMF software only requires object files, applications provided by third parties can be configured to use NonStop AutoTMF software.
- Because NonStop AutoTMF software does not alter the basic processing by application programs, your application execution procedures require no change. Changes to operational procedures to manage the audited database may be required.
- Through use of proprietary technology, NonStop AutoTMF software will perform many database I/Os in each transaction. Accesses to multiple files are managed within a single transaction. NonStop AutoTMF software can duplicate the performance of programs that explicitly manage transactions.
- If your application already manages transactions for an audited database, you can use NonStop AutoTMF software to audit any other files that are currently unaudited.

- Because NonStop AutoTMF software provides transactions only when needed, you can migrate your application, incrementally, to one that manages business-level transactions for maintaining a consistent database.

Capabilities of NonStop AutoTMF Software

NonStop AutoTMF software allows you to audit your data without reprogramming application programs so that you can take advantage of the following capabilities of audited data:

- Data replication for disaster recovery using the HP NonStop Remote Database Facility (RDF) product.
- Multi-platform data replication using third-party products that read audit trail records.
- Online backup and file/volume recovery to provide data protection for active databases in a continuously available application. Backups during offline processing can be eliminated. Files can be restored to the point of the last committed update.
- Performance improvements through optimized processing for audited files. These optimizations include update buffering and block split I/O elimination; these optimizations can make significant reductions in physical I/O operations to reduce disk queuing and response time.

NonStop AutoTMF software provides an incremental migration path to upgrade an application to manage business-level transaction protection and recovery.

NonStop AutoTMF software is easy to install and manage. NonStop AutoTMF software is not privileged and does not require a SYSGEN or the use of the SUPER user.

Limitations

In the current version, NonStop AutoTMF software does not support the generation of transactions for audited SQL tables.

2 Installation and Basic Management

Prerequisites

Read the softdoc before you install NonStop AutoTMF software.

NonStop AutoTMF software generates transactions on behalf of application programs. The TMF product views NonStop AutoTMF software like any other transactional program. TMF must be installed and configured before using NonStop AutoTMF software.

Before using NonStop AutoTMF software on a production system, you should consider the additional transaction activity generated by the application programs.

Volumes containing the database files and the NonStop AutoTMF configuration must be specified in the TMF configuration as audited data volumes.

HP NonStop SQL/MP must be initialized before using NonStop AutoTMF software, and the SQL/MP system catalog must be created. You will not require a separate license for SQL/MP.

For more information about the configuration and operation of the TMF product, refer to the following HP documentation: the *TMF Planning and Configuration Guide*, the *TMF Operations and Recovery Guide* and the *TMF Reference Manual*.

For special security considerations pertaining to your environment, see Security and Availability on page A-18.

Installing NonStop AutoTMF Software

Have your NonStop AutoTMF software licensing instructions available. Consult the softdoc file for any changes to the installation procedures. Then proceed as follows:

1. Move Files from the Product Media to the Installation Subvolume.
2. Install NonStop AutoTMF Software Files.
3. (Optional) Start an EMS distributor.
4. Create the System Database
5. Install the NonStop AutoTMF Software License.
6. Create the Mapping Database.
7. Update System Coldload Procedures.
8. Install EMS templates.
9. If programs use a user library, bind the NonStop AutoTMF runtime library into your existing libraries. See Combining a User Library with the NonStop AutoTMF Runtime on page 3-8 in Section 3, Preparing Programs.

Move Files from the Product Media to the Installation Subvolume

The NonStop AutoTMF product media contains the files listed in Table 2-1. Instructions for placing the NonStop AutoTMF software in the ISV subvolume ZAUTOTMF are delivered on the distribution media. After the initial installation, these files can also be copied from another system on your network, or copied using FTP over the Internet from an HP site.

The subvolume ZAUTOTMF contains the following files:

Table 2-1. ZAUTOTMF Subvolume

FIle Name	Contents
ATMFPAK	A PAK file containing the AutoTMF product files listed in Table 2-2
INSTALL	TACL macro for installing AutoTMF
SESCTMPL	Source EMS templates
T0581H*nn*	Softdoc where *nn* designates the PVU level
ZESCTMPL	EMS templates

The actual operational product files for NonStop AutoTMF software are listed in Table 2-2. Note that the product subvolume contains versions of the runtime library for

all NonStop servers. The INSTALL macro determines which runtime library files to install depending on the processor type of the server.

Table 2-2. NonStop AutoTMF Product Files in ATMFPAK

File Name	Contents
ESCERROR	Error and warning messages
ESCFLTR	EMS filter for NonStop AutoTMF software
ESCHELP	Help text for command interpreter ESCORT (ESCORT CI)
ESCORT	NonStop AutoTMF command interpreter
ESCMON	Monitor program
ESCRUNDL	NonStop AutoTMF TNS/E runtime DLL
ESCRUNN	Linkable NonStop AutoTMF TNS/R runtime user library for PTAL programs
ESCRUNNL	Linkable NonStop AutoTMF TNS/R runtime user library for COBOL and C programs
ESCRUNNM	Executable NonStop AutoTMF TNS/R runtime user library for COBOL and C programs
ESCRUNNT	Executable NonStop AutoTMF TNS/R runtime user library for PTAL programs
ESCRUNTM	NonStop AutoTMF TNS runtime user library
UPDDLLS	TACL macro to update rebased public DLLs in AutoTMF subvolume (On Integrity NonStop severs and NonStopTM BladeSystem only)
ZESCTMPL	EMS template file

Install NonStop AutoTMF Software Files

INSTALL restores the product files from ZAUTOTMF to the operational subvolume; the default operational subvolume is $SYSTEM.ESCORT. Do not install the files in $SYSTEM.SYSTEM or $SYSTEM.SYSnn.

To install AutoTMF in $SYSTEM.ESCORT, type the following TACL commands:

```
VOLUME $<vol>.ZAUTOTMF
RUN INSTALL
```

To install AutoTMF in a subvolume other than $SYSTEM.ESCORT, specify the subvolume when you run INSTALL:

```
RUN INSTALL $TOOLS.AUTOMF
```

In a network of systems, where application software might be copied from one system to another, choose a volume and subvolume that can be specified on all systems.

TNS and TNS/R Installation

In the example below, AutoTMF is installed in subvolume $TOOLS.AUTOTMF:

```
24> install $tools.autotmf
HP Nonstop(tm) AutoTMF(tm) Software File Installer

UNPAK - File decompression program - T1255G06 - (2008-06-03)

Archive version: 1
File Mode RESTORE Program - T9074G08 (21JUL2008) (AFO)
(C)2000 Compaq (C)2006 Hewlett Packard Development Company, L.P.
Drives: (\SIERRA.$Y52P)
System: \ATOM  Operating System: H06  Tape Version: 3
Backup options:  NO AUDITED, BLOCKSIZE 8, NO IGNORE, NO OPEN, PARTONLY OFF,
                 INDEXES IMPLICIT
*WARNING-7147*  Files created and stored via OSS and SQL/MX objects are not
                 supported.
Restore time: 11Apr2011 17:13  Backup time: 11Apr2011 17:08           Page: 1

Tape: 1       Code             EOF      Last modif   Owner RWEP  Type Rec Bl

$TOOLS.AUTOTMF
 ESCERROR                    274432  1Apr2011 11:20 170,46  NCNC  K    128  4
 ESCFLTR       101              334  9May2008  9:34 170,46  NCNC
 ESCHELP                    1544192  1Apr2011 11:53 170,46  NCNC  K   1928  4
 ESCMON        100          6318080 11Apr2011 17:08 170,46  NCNC
 ESCORT        100          5413732 11Apr2011 17:08 170,46  NCNC
 ESCRUNDL      800         10708648  7Apr2011 16:43 170,46  NCNC
 ESCRUNN       700          3240720  7Apr2011 16:43 170,46  NCNC
 ESCRUNNL      700          5704456  7Apr2011 16:43 170,46  NCNC
 ESCRUNNM      700          6014424  7Apr2011 16:43 170,46  NCNC
 ESCRUNNT      700          3376488  7Apr2011 16:43 170,46  NCNC
 ESCRUNTM      100          6742016 11Apr2011 17:05 170,46  NCNC
 ZESCTMPL      839            57344  7Apr2011 16:42 170,46  NCNC  K    510  4

Summary Information

Files restored = 12  Files not restored = 0

AutoTMF software file installation is complete.
Proceed to installation instructions described
in the user manual.
```

TNS/E Installation

On a TNS/E server, in addition to restore of the product files, the INSTALL macro performs the following operations to set up the procedure call intercepts for the native COBOL and C language runtime DLLs:

1. Finds the current public language runtime DLLs required to support native COBOL and C programs
2. Using the native mode linker ELD, moves a copy of the public DLLs into the AutoTMF operational subvolume and “rebases” the public DLLs to the AutoTMF subvolume.
3. Prepares the DLLs.

See section TNS/E Host Language Runtime DLLs in Appendix A, System Management for further details about rebasing public DLLs.

In the example below, NonStop AutoTMF software is installed on $TOOLS.AUTOTMF:

```
71> install $tools.autotmf
HP Nonstop(tm) AutoTMF(tm) Software File Installer

UNPAK - File decompression program - T1255H01 - (2009-09-25)

Archive version: 1
File Mode RESTORE Program - T9074H01 (04FEB2009) (AFN)
(C)2000 Compaq (C)2007 Hewlett-Packard Development Company, L.P.
Drives: (\ATOM.$Y6JC)
System: \ATOM  Operating System: H06  Tape Version: 3
Backup options:  NO AUDITED, BLOCKSIZE 8, NO IGNORE, NO OPEN, PARTONLY OFF,
                 INDEXES IMPLICIT
*WARNING-7147*  Files created and stored via OSS and SQL/MX objects are not
                supported.
Restore time: 11Apr2011 17:13  Backup time: 11Apr2011 17:08          Page: 1

Tape: 1       Code             EOF      Last modif   Owner RWEP  Type Rec Bl

$TOOLS.AUTOTMF
 ESCERROR                     274432  1Apr2011 11:20 170,46  NNNN  K    128  4

 ESCFLTR       101              334  9May2008  9:34 170,46  NNNN
 ESCHELP                    1544192  1Apr2011 11:53 170,46  NNNN  K   1928  4

 ESCMON        100          6318080 11Apr2011 17:08 170,46  NNNN
 ESCORT        100          5413732 11Apr2011 17:08 170,46  NNNN
 ESCRUNDL      800         10708648  7Apr2011 16:43 170,46  NNNN
 ESCRUNN       700          3240720  7Apr2011 16:43 170,46  NNNN
 ESCRUNNL      700          5704456  7Apr2011 16:43 170,46  NNNN
 ESCRUNNM      700          6014424  7Apr2011 16:43 170,46  NNNN
 ESCRUNNT      700          3376488  7Apr2011 16:43 170,46  NNNN
 ESCRUNTM      100          6742016 11Apr2011 17:05 170,46  NNNN
 UPDDLLS       101             5942  8Apr2011 14:02 170,46  NNNN
 ZESCTMPL      839            57344  7Apr2011 16:42 170,46  NNNN  K    510  4


Summary Information

Files restored = 13  Files not restored = 0

Rebase and Prepare Public DLLs...

$system.ZDLL018.ZOSSKDLL rebased to $tools.autotmf.ZOSSKDLL and prepared
$system.ZDLL018.ZICNVDLL rebased to $tools.autotmf.ZICNVDLL and prepared
$system.ZDLL018.ZI18NDLL rebased to $tools.autotmf.ZI18NDLL and prepared
$system.ZDLL018.ZCRTLDLL rebased to $tools.autotmf.ZCRTLDLL and prepared
$system.ZDLL018.ZCREDLL rebased to $tools.autotmf.ZCREDLL and prepared
$system.ZDLL018.ZCOBDLL rebased to $tools.autotmf.ZCOBDLL and prepared

AutoTMF software file installation is complete.
Proceed to installation instructions described
in the user manual.
```

Start an EMS distributor

Before you install the NonStop AutoTMF EMS templates in the system templates, you can view NonStop AutoTMF messages through an EMS distributor, using the NonStop

AutoTMF templates and filter. Enter the following TACL commands in a separate terminal window:

```
ADD DEFINE =_EMS_TEMPLATES, CLASS MAP, FILE ZESCTMPL

EMSDIST TYPE P, COLLECTOR $0, TEXTOUT $HOME, FILTER ESCFLTR
```

The EMS distributor will display NonStop AutoTMF events.

Create the System Database

The System Database (SysDB) is created using ESCORT CI. When starting ESCORT CI during the initial installation, a warning message appears directing you to create a SysDB.

1. Invoke ESCORT by using a RUN command:

```
19> run $tools.autotmf.escort
HP Nonstop(tm) AutoTMF(tm) Command Interpreter(T0581V03) - System \RDF10
Copyright Hewlett-Packard Company 2000-2002
Copyright Carr Scott Software Incorporated 1996-2002

*Warning* The SysDB cannot be found. Issue the CREATE SYSDB command to
* 1601  * create it. Enter HELP CREATE-SYSDB; for more information.
```

2. Then issue the CREATE command:

   CREATE SYSDB [ON *volume*];

```
AutoTMF 1? create sysdb;
--- SQL Catalog $SYSTEM.ESCCATLG is being created.
--- SQL Catalog $SYSTEM.ESCCATLG has been created.
--- SysDB table $SYSTEM.ESCSYSDB.MAPDBS created.
--- SysDB table $SYSTEM.ESCSYSDB.REGISTRY created.

* Error * The monitor process $ZESC is not executing.
*  652  * To start a monitor, issue the START MONITOR command.

*Warning* Process $ZESC is not configured as a monitor process.
* 1806  * The CREATE MAPDB command configures monitor processes.
*       * Enter HELP CREATE-MAPDB; for more information.
```

volume is the audited volume for SysDB. The default volume for SysDB is $SYSTEM. If creating tables on $SYSTEM is difficult due to system security or if $SYSTEM is not audited, specify an alternative volume. SysDB may be created on any audited volume.

Note that ESCORT CI is looking for the AutoTMF monitor $ZESC. The monitor will be configured and started in step Create the Mapping Database below, after the AutoTMF software license has been installed in the next step.

See CREATE SYSDB on page 6-54 for more details on this command, and specify the SECURE option as indicated in that discussion to control access to users who are authorized to alter the NonStop AutoTMF global configuration.

SQL tables must be registered in an SQL catalog. ESCORT CI automatically creates an SQL catalog on the volume specified in the CREATE SYSDB command, in a

subvolume called ESCCATLG. Use the CATALOG option to specify a different catalog, as shown in the details under CREATE SYSDB on page 6-54.

Install the NonStop AutoTMF Software License

Follow the licensing procedure described in the documentation that accompanies your NonStop AutoTMF software. The license is installed by using an Escort CI command. If you move the NonStop AutoTMF software to a different system, you must request a license for that system.

If you are upgrading your NonStop AutoTMF software license from a limited license to a permanent license, simply install the new license received from the License Manager. Installing the new license is not disruptive and does not require any interruption of the running application.

To display the status of your NonStop AutoTMF software license, use the Monitor STATUS LICENSE command after the next installation step (that is, after MapDB has been created and the Monitor process is started).

Note. On H-series and J-series systems, product licensing is based on the configured number of logical CPUs. If you add logical CPUs to your system, contact the HP license manager to obtain a new license that accounts for the increased number of CPUs.

Create the Mapping Database

The Mapping Database (MapDB) is a collection of SQL tables that store NonStop AutoTMF configuration information. To create MapDB, use the command:

```
CREATE MAPDB [ON volume];
```

MapDB tables are created in subvolume ESCMAPDB; the default volume is the SysDB volume. As in the case of SysDB, the MapDB volume specified must be an audited volume.

```
AutoTMF 4? create mapdb;
--- MapDB table $SYSTEM.ESCMAPDB.REGISTRY created.
--- MapDB table $SYSTEM.ESCMAPDB.ATMFATTR created.
--- MapDB table $SYSTEM.ESCMAPDB.ATMFPROG created.
--- Map Database for AutoTMF created.
--- SysDB updated
--- Changes will take effect when the Monitor is restarted.
--- Starting Monitor process $JTMF

AutoTMF Monitor 1.9.1 - 09JUN2011 -- $ZESC (4,849) - System \ATOM
Started at May 11 2011 16:15:57, elapsed time = 0:00:00
Backup process $ZESC (5,1010), no takeovers
Monitor hometerm: $0
EMS event logging level is NORMAL.
AutoTMF License Information
License is                  Valid
Maximum logical processors   8
```

When you create MapDB, the NonStop AutoTMF monitor process is started to manage access to MapDB. See CREATE MAPDB on page 6-52 for details about this command.

When the monitor process is started, the following messages should be displayed by the EMS distributor:

```
    15:58 11MAY11 099,05,093 $ZESC 100 ESCMON starting Version
                                 1.9.1 - 09JUN2011
15:58 11MAY11 099,05,093 $ZESC 149 ESCMON message
                                 AutoTMF License Information
                                 License is                  Valid
                                 Expiration date:            Oct 15 2011
                                 Maximum logical processors  8
15:58 11MAY11 099,05,093 $ZESC 149 ESCMON message
                                 AutoTMF Global Settings
                                 AutoTMF            ON
                                 ATMFCommonTx       ON
                                 ATMFMaxTime        16 seconds
                                 ATMFMaxUpdate      32
                                 ATMFNowait         OFF
                                 ATMFIsolation      Weak
                                 ATMFTxHoldoff      0 seconds
                                 ATMFAuditRename    OFF
                                 ATMFSkipNullRecs   OFF
                                 ATMFReadThruLock   ON
                                 ATMFAutoCommit     115 minutes
                                 ATMFTXTimeout      OFF (requires TMF 3.6)
                                 ATMFAbendNoAudit   ON
                                 ATMFOptmzUnlocks   OFF
                                 ATMFSTOPonTMFErr   OFF
                                 ATMFMaxTX          100
                                 ATMFNoWarnLongTx   OFF
                                 ATMFSeparateTx     OFF
                                 MaxMonitorOpens    512
                                 DynamicTrace       ON
                                 SecureTrace        ON
                                 EMSCollector       $0
15:58 11MAY11 099,05,093 $ZESC 101 ESCMON started Version
                                 1.9.1 - 09JUN2011
15:58 11MAY11 099,05,093 $ZESC 112 ESCMON ATMFFILE table cached. Table
                                 entry count = 2.
15:58 11MAY11 099,05,093 $ZESC 112 ESCMON ATMFPROG table cached. Table
                                 entry count = 2.
15:58 11MAY11 099,05,093 $ZESC 104 ESCMON backup started, cpu 4
```

The monitor process is a fault-tolerant process-pair that has the process name $ZESC; you should not have to restart it unless you intentionally stop it or cold load the system. If you discover that the monitor process is not running, start it by using the command:

```
START MONITOR;
```

You may wish to configure the NonStop AutoTMF monitor process parameters, such as priority and cpus. In particular, the monitor priority should be higher than all application programs to avoid delays during process startup.

The section Monitor Process on page A-2 describes, in detail, how to configure and determine the status of the Monitor process.

Update System Coldload Procedures

Once you alter the database files to be audited, the NonStop AutoTMF monitor process becomes a critical component of application availability. You must ensure that the monitor process is started whenever system maintenance has required it to be stopped. You should include the start of the monitor process with the other operational steps that normally follow a system cold load.

You should start the monitor process after TMF has been started but before starting Pathway or any application processes. Start the NonStop AutoTMF monitor process by using the following TACL command:

```
RUN $TOOLS.AUTOTMF.ESCORT START MONITOR;
```

The user ID required to start the monitor also must have read access to SysDB and MapDB tables.

Install EMS templates

The file ZESCTMPL contains EMS templates for all events generated by the NonStop AutoTMF software. Please refer to the *DSM Template Services Manual* for instructions about installing system templates.

You can use the EMS filter file ESCFLTR with an EMS distributor to display NonStop AutoTMF events.

Updating NonStop AutoTMF Software

1. Stop application programs and all processes that use the NonStop AutoTMF runtime library. In this case, display (and optionally stop) all programs that use the NonStop AutoTMF runtime library by using the INFO LIBRARY command.
2. Stop the NonStop AutoTMF monitor process by using the command STOP MONITOR.
3. Replace the NonStop AutoTMF software files as described above under Install NonStop AutoTMF Software Files.
4. If required, bind your application user libraries with the new NonStop AutoTMF runtime library. See Combining a User Library with the NonStop AutoTMF Runtime in Section 3, Preparing Programs.
5. If you need to preserve modification timestamps or preserve license attributes of object files when preparing programs, perform the necessary steps to make the new Escort CI a licensed program, as described in Preserving the Modification Timestamp of Object Files or in Preserving License Attribute when Preparing Privileged Programs in Section 3, Preparing Programs.
6. Install the new NonStop AutoTMF EMS templates.

7. Restart the NonStop AutoTMF monitor process.
8. Restart the application programs.

Running Different Versions of NonStop AutoTMF on the Same System

If you need to run different versions of NonStop AutoTMF on the same system, for example to test a newer version of the product, you must configure an alternate NonStop AutoTMF software environment (Monitor and MapDB) and install the test software in a different location than the subvolume currently in use. For details on configuring and using an alternate environment, see Configuring and Using an Alternate NonStop AutoTMF Environment on page A-4 in Appendix A, System Management.

Upgrading the Operating System

For systems running only TNS or TNS/R applications, an upgrade to the Operating System does not require any specific updates to NonStop AutoTMF.

On Integrity NonStop servers or NonStop Blades running TNS/E applications, you must update the versions of the DLLs that were rebased and prepared at the time the current version of NonStop AutoTMF was installed. To do so, you can run the UPDDLLS macro found in the AutoTMF operational subvolume. See Running UPDDLLS on page A-16 in Appendix A, System Management for further details.

Disabling NonStop AutoTMF Software

To disable NonStop AutoTMF software and run application programs, use any one of the following steps:

- Alter the database files to unaudited. If no files are audited, the NonStop AutoTMF runtime library will not perform any extra operations to manage automatic transactions.
- Stop the NonStop AutoTMF monitor process. If the monitor process is not available to supply NonStop AutoTMF configuration information, the NonStop AutoTMF runtime library will revert to the passthrough mode and will not perform any extra operations to manage automatic transactions.
- Alter the NonStop AutoTMF global configuration to disable automatic transactions. The command is:

  ```
  ALTER GLOBAL AUTOTMF OFF;
  ```

Removing NonStop AutoTMF Software

1. If you have prepared and bound your application's user libraries with the NonStop AutoTMF runtime library, restore those user libraries with copies that are unprepared and unbound.
2. Use the UNPREPARE command to restore the application program object files to their original state.
3. Stop the NonStop AutoTMF monitor process.
4. Use SQLCI to purge the SysDB and MapDB tables.
5. Use SQLCI to drop any catalog created during installation.
6. Update your startup and shutdown procedures to remove the commands that start and stop the monitor process.

Shortcut to Run ESCORT CI

NonStop AutoTMF software is placed in its own subvolume, not $SYSTEM.SYSTEM. You can create a shortcut for the RUN command that allows you to type ESCORT at the TACL prompt. Assuming NonStop AutoTMF is installed in the default subvolume $system.escort, here are two suggestions:

- Create an EDIT file $SYSTEM.SYSTEM.ESCORT that contains the following two lines of text:

```
?TACL MACRO
RUN $SYSTEM.ESCORT.ESCORT %*%
```

- Update the $SYSTEM.SYSTEM.TACLLOCL file, or your own TACLCSTM file, by inserting the following line:

```
[#DEF ESCORT ALIAS|BODY|$SYSTEM.ESCORT.ESCORT]
```

3 Preparing Programs

Overview

Before preparing any programs, ensure that you have a working version of the application to be prepared. Read the suggestions at the end of this section before preparing any programs.

You must prepare programs to use NonStop AutoTMF software. Preparation changes object files to invoke the NonStop AutoTMF runtime library instead of making direct calls to operating system procedures.

You prepare programs by using the PREPARE command, which performs two operations on each object file:

- Retargets selected system procedure calls to the NonStop AutoTMF runtime library.
- For TNS and TNS/R programs, either sets the NonStop AutoTMF runtime library as the program's user library, or verifies that an existing user library contains the NonStop AutoTMF runtime.
- For TNS/E programs, sets the NonStop AutoTMF runtime DLL as a DLL to the program or to the program's user library.

You do not need the source code of the programs, nor do you need to compile the programs in any special manner. However, you do need to prepare a program whenever that program is recompiled.

Object code is not changed by NonStop AutoTMF. From the viewpoint of the application programmer, tester, or debugger, the execution of prepared and unprepared programs is the same.

Preparation changes the object-file modification timestamp to the time of preparation. To preserve the modification timestamp, see Preserving the Modification Timestamp of Object Files below.

Preparation does not invalidate object-file acceleration or SQL compilation, nor does preparation change binder or linker attributes such as HIGHPIN, HIGHREQUESTERS, INSPECT, and so on, or the Binder time stamp or the size of the object file.

However, preparation removes LICENSE attribute of an object file, unless one of 2 conditions are met:

- the PREPARE is performed by SUPER.SUPER, or
- the Escort CI procedure CALLABLE^SET^LICENSE is made callable, as described in Preserving License Attribute when Preparing Privileged Programs below.

The NonStop AutoTMF runtime user library or DLL contains routines that intercept the retargeted system calls. User libraries or DLLs do not increase the size of the object file and are shared in virtual memory by all processes that use them. To upgrade versions of NonStop AutoTMF software, the runtime library or DLL may be replaced whenever the program is not running.

If your application programs have user libraries, you should prepare these user libraries first, then prepare the application programs. The preparation of a program will fail if the referenced user library has not been prepared; this arrangement eliminates many errors that might occur if you could prepare a program that referenced an unprepared user library.

Preparing Programs

Using the PREPARE Command

The PREPARE command processes an object file and updates the procedure identifier lists to rename references to selected system procedures, such as READ and FILEINFO, and to NonStop AutoTMF runtime library procedures, such as R^^D and F^L^^NFO. Renamed procedure names are the same length as the corresponding file system procedure name, but have all internal vowels changed to the caret (^) character. Use INFO PREPARE to display a complete list of names.

The PREPARE command will process collections (that is, file sets) of object files. To use PREPARE, you must have write access to the object files. The command changes the object files, so you may wish to retain backup copies of the object files until you have tested the prepared programs.

The following sample terminal session shows PREPARE used on a single object file:

```
9> escort
HP Nonstop(tm) AutoTMF(tm) Command Interpreter(T0581V03) - System \RDF10
Copyright Hewlett-Packard Company 2000-2004
Copyright Carr Scott Software Incorporated 1996-2004
AutoTMF 1? prepare data.myobj.batchobj;

--- $DATA.MYOBJ.BATCHOBJ preparation complete
--- AutoTMF runtime library $SYSTEM.ESCORT.ESCRUNTM
AutoTMF 2?
```

Use the INFO PROGRAM command to verify the state of the object file, for example:

```
AutoTMF 7? info program myobj.batchobj,detail;

$DATA.MYOBJ.BATCHOBJ              Apr 29 2004, 14:07
    Accelerated TNS Program
    User Library $SYSTEM.ESCORT.ESCRUNTM
    Prepared
    Intercept proc calls prepared: 33
    Attributes: Inspect, Saveabend
    Binder timestamp: 23Apr04  8:27
    AXCEL timestamp:  23Apr04  8:27
```

If your application consists of many programs, you can prepare entire subvolumes through one command; however, you should keep a record of the preparation by using the LOG command, for example:

```
AutoTMF 1? log to srvobj.preplog;
AutoTMF 2? prepare srvobj.*;
<< Messages for each program >>
Object file summary: prepared = 72, unprepared = 3, excluded = 4
AutoTMF 3? log stop;
```

The PREPARE command processes only object files and ignores all other files. If some files are not prepared because of errors, correct the problem, then re-issue the PREPARE. You may prepare a program as often as you wish.

To view the prepared status of a collection of object files, use INFO PROGRAM. For example:

```
AutoTMF 1? info program $data.srvobj.*;
$DATA.SRVOBJ

         Type        SQL        User Library               Status
...
EGETEMP  TNSE Loadfil                                      Not Prepared
OGETEMP  TNS  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
OGETINFO TNS  Program                                      Not prepared
OPNUPD   AXL  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
ORDEMP   AXL  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
OREQ     TNS  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
OSERV    TNS  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
RGETEMP  TNSR Loadfil          $TOOLS.ESCORT.ESCRUNNT      Prepared
TSTATMFO TNS  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
XLGETEMP AXL  Program          $TOOLS.ESCORT.ESCRUNTM      Prepared
XREQ     TNSR Loadfil          $TOOLS.ESCORT.ESCRUNNT      Prepared
XSERV    TNSR Loadfil          $TOOLS.ESCORT.ESCRUNNT      Prepared
YGETEMP  TNSE Loadfil                                      Not prepared
YREQ     TNSE Linkfil                                      Not prepared
YSERV    TNSE Linkfil                                      Not prepared

Object file summary: prepared = 10, not prepared = 5, excluded = 0
```

Preparing TNS Programs

The NonStop AutoTMF runtime library for TNS programs (ESCRUNTM) contains prepared copies of the COBOL85, C, CRE, and GPLIB runtime libraries. This arrangement is required to allow NonStop AutoTMF software to intercept all file system

calls made by those libraries. These libraries also exist in the “system library” created by SYSGEN, but you cannot intercept calls that are made to the system library.

If a TNS program currently calls a procedure in the system library routine at run time, the calls are automatically retargeted to the ESCRUNTM user library if the ESCRUNTM user library contains the same procedure.

Any COBOL, C, or TAL TNS program can use the ESCRUNTM library. If the program does not call a language runtime routine, no side-effects occur when the routine is bound to the runtime library.

Preparing TNS/R Programs

Preparation steps for TNS/R programs are the same as described above, with a few limitations.

For TNS/R programs, the language runtimes have a significantly different implementation:

- The NonStop OS supports special initialization routines. If the program or library contains procedures that have special names, those procedures run automatically before the program starts. For example, the C runtime library has an initialization routine to initialize the heap data structures.
- TNS/R language runtime libraries are implemented in Shared Runtime Libraries (SRLs). When a TNS/R object is made executable, each external reference is assigned to the system library, a user library, or to a system SRL. Once a reference is assigned to an SRL, the reference cannot be changed.
- SRLs are independent of one another and do not share data unless explicitly declared; several system SRLs, notably the C and C++ SRLs have conflicting definitions and cannot be linked together.

These features of TNS/R mode enhance the capabilities of the language runtime support but also make interception of procedure calls more complex, resulting in the requirements and limitations described below.

ESCORT CI identifies three classes of TNS/R programs and deals with them as described in the following table:

Table 3-1. TNS/R Program Classes and NonStop AutoTMF Libraries:

Program Class	Runtime library	Linkable library	Preparation Processing
COBOL/C	ESCRUNNM	ESCRUNNL	SRLs references eliminated; all calls intercepted in NonStop AutoTMF UL.
pTAL	ESCRUNNT	ESCRUNN	Direct Enscribe/GPLIB calls intercepted in NonStop AutoTMF UL
C++/Other SRL	ESCRUNNT	ESCRUNN	Direct Enscribe/GPLIB calls intercepted in NonStop AutoTMF UL; all other calls not intercepted

COBOL and C Programs

Before they are prepared, COBOL and C programs reference the COBOL and/or C SRLs as well as the CRE library routines. The COBOL and C programs automatically have compiler-defined data structures that are referenced by the initialization routines in those SRLs.

If a program references the COBOL/C/CRE SRLs (and no others), the PREPARE command will automatically retarget calls made to those libraries to the ESCRUNNM user library provided as part of the NonStop AutoTMF TNS/R support. The prepared program will have no references to SRLs.

ESCRUNNM is an executable user library that contains the COBOL, C, CRE, and GPLIB runtime libraries. The COBOL/C/CRE libraries are compatible and use the data structures defined by the CRE. Programs must have data structures that are compatible with the CRE initialization routines.

ESCRUNNL is a linkable version of the ESCRUNNM library; use this object if you relink the NonStop AutoTMF runtime with your own TNS/R user library.

pTAL Programs

Programs written exclusively in pTAL do not reference any SRL. (A pTAL program that uses the CRE SRL will be prepared in the same way as a COBOL or C program.)

ESCRUNNT is an executable user library that contains only the GPLIB libraries. Specifically, ESCRUNNT does not contain initialization routines for CRE data structures. If a program gets a “heap error” during initialization the program probably has no CRE data structures, and is using the CRE SRL.

ESCRUNN is a linkable version of the ESCRUNNT library; use this object if you relink the NonStop AutoTMF runtime with your own TNS/R user library.

C++ Programs

C++ programs reference both the C++ and C SRLs. However, the C++ and C libraries have conflicting externals and cannot be linked together in the same library; therefore, a NonStop AutoTMF user library cannot intercept calls from C++ programs.

If a program references the C++ SRL, the PREPARE command:

- Will direct calls to Enscribe, causing GPLIB to be intercepted by the ESCRUNNT user library.
- Will not modify calls made to any SRL (including COBOL, C, or CRE). Thus, database calls made through SRLs will not be intercepted. If the program uses COBOL statements to access audited files, error 75 usually results.

When such a program is prepared, ESCORT CI issues a warning:

```
AutoTMF 1? prepare srvobj.myCplusp;

*Warning* $DATA.SRVOBJ.MYCPLUSP references a set of system SRLs
* 1665  * that may not be compatible with the AutoTMF runtime library.
*       *
*       * The program will be prepared to intercept only direct calls
*       * to Enscribe procedures and not to calls made through the
*       * COBOL, C or CRE runtime libraries.
*       *
--- $DATA.SRVOBJ.MYCPLUSP preparation complete
--- AutoTMF User Library $DATA.ESCORT.ESCRUNNT
```

Programs referencing other SRLs

If you have programs that reference other SRLs, such as the TCP/IP library ZINETSRL, the PREPARE command will operate as it does for C++; that is, COBOL, C, and CRE database calls will not be intercepted.

PREPARE can cause all database calls to be intercepted, if you link a prepared copy of the SRL library code with the program (eliminating the need to use the SRL); however, you should contact product support for assistance in obtaining linkable versions of SRLs (Known as “RELs” instead of SRLs).

Preparing TNS/E Programs

In TNS/E programs, the retargeting of system procedure calls is performed in the same manner as it is for TNS and TNS/R objects.

However, since the NonStop AutoTMF runtime for TNS/E programs is implemented as a DLL instead of as a user library, the PREPARE command simply sets the NonStop

AutoTMF runtime DLL (ESCRUNDL) as a DLL of the program. The example below shows the preparation of program $data.testobj.egetemp:

```
AutoTMF 3? prepare egetemp;
--- $DATA.TESTOBJ.EGETEMP preparation complete
--- AutoTMF DLL $TOOL.AUTOTMF.ESCRUNDL
AutoTMF 4? info program egetemp,detail;

$DATA5.JOJOTEST.EGETEMP              May 6 2005 , 10:09
    Executable TNS/E Program
    Not Preset; Nonempty Liblist
    DLLs: escrundl
    Pre-public DLL path:  $TOOL.AUTOTMF
    Intercept proc calls prepared: 9
    Attributes: Highpin, Highrequesters, Visual Inspect
    ELD Create timestamp       :  6May05 10:06
    ELD/Escort Update timestamp:  6May05 10:06
```

Preparing Programs that Have a User Library

If you attempt to prepare a program that references a user library, the following error message may be issued:

```
AutoTMF 1? prepare srvobj.applobj;
* Error * The user library $DATA.LIB.TINFO does not
*  456  * contain the AutoTMF runtime library.
*       * Object file $DATA.SRVOBJ.APPLOBJ will not be prepared.
```

This error occurs because a program cannot be successfully prepared if it references a user library that does not contain the NonStop AutoTMF runtime library.

If any of your programs refer to a user library, the user library should be prepared.

TNS and TNS/R Programs

Once the user library has been prepared, the NonStop AutoTMF runtime library must be bound with the user library object file. This step must be completed before you prepare the programs that reference the user library. The preparation of such programs verifies that the user library contains the NonStop AutoTMF runtime library and if the NonStop AutoTMF runtime library is not present, the prepare operation fails.

The PREPARE command does not alter an existing reference to a user library, so you must replace each existing user library with a user library that contains the NonStop AutoTMF runtime library. You should not add your user library routines to the standard $SYSTEM.ESCORT.ESCRUNTM object file.

TNS/E Programs

A TNS/E program can have a user library and multiple DLLs. A user library can have its own DLLs. Similarly, a DLL can have its own DLLs. This feature of DLLs eliminates the need to bind or link the AutoTMF runtime into the user library. Instead, when the user library is prepared, the NonStop AutoTMF software runtime DLL is set as a DLL to the user library.

Preparing a User Library

By default, the PREPARE command only modifies executable object files. If you attempt to prepare a program that references a user library, the following message is displayed:

```
AutoTMF 1? prepare $DATA.LIB.TINFOB;

*Warning* Object file $DATA.LIB.TINFOB has no MAIN procedure.
* 1658  * A user library must be prepared with the USERLIB option
*       * and then bound with the AutoTMF runtime library.
```

To prepare a user library, use the PREPARE command specifying the USERLIB option as follows:

```
AutoTMF 1? prepare $DATA.LIB.TINFOB, USERLIB;

--- $DATA.LIB.TINFOB preparation complete, but has no MAIN;
--- a user library must be bound with the AutoTMF runtime library.
```

Intercept Libraries

Certain third-party software products use a user library to intercept NonStop OS procedure calls, much like NonStop AutoTMF software does. In that case, the user library should not be prepared prior to binding the NonStop AutoTMF runtime to the user library. If a user library contains both intercept procedures and ordinary application procedures, special processing is required; contact product support for assistance.

Combining a User Library with the NonStop AutoTMF Runtime

TNS Library

1. Prepare the application user library object using the PREPARE command:

   ```
   [run] $SYSTEM.ESCORT.ESCORT PREPARE myuserlb, UL;
   ```

2. BIND the application user library with the runtime object file ESCRUNTM, as follows:

   ```
   BIND
   @SELECT LIST * OFF
   @ADD * FROM myuserlb
   @ADD * FROM $SYSTEM.ESCORT.ESCRUNTM
   @SET LIKE $SYSTEM.ESCORT.ESCRUNTM
   @SET HIGHPIN ON
   @SET HIGHREQUESTERS ON
   @BUILD myuserlb!
   ```

3. Accelerate the new user library. Specify the UL option, as follows:

- For NonStop S-series servers:

```
AXCEL myuserlb,myuserlb; UL
```

- For NonStop Integrity servers:

```
OCA myuserlb; UL
```

4. If the NonStop AutoTMF EMS message template file ZESCTMPL has not been installed, move a copy of ZESCTMPL into the same subvolume as the user library. It might be needed to generate messages to the home terminal.

TNS/R Library

1. Create or locate a linkable version of the user library.
2. Prepare the application user library object using the PREPARE command:

```
[run] $SYSTEM.ESCORT.ESCORT PREPARE mynatlb, UL;
```

3. Use nld to combine ESCRUNN and ESCRUNNL into two versions of the user library corresponding to the two classes of TNS/R programs:

- For use with PTAL and C++ TNS/R programs:

```
nld ESCRUNN mynatlb -o mynatlbt -ul
```

- For use with COBOL and C TNS/R programs:

```
nld ESCRUNNL mynatlb -o mynatlbc -ul
```

4. If the NonStop AutoTMF EMS message template file (ZESCTMPL) has not been installed, move a copy of it into the same subvolume as the user library. ZESCTMPL might be needed to generate messages to the home terminal.

TNS/E Library

Combining the user library with the NonStop AutoTMF runtime is not required for TNS/E user libraries or DLLs. Preparing the user library or user DLL establishes the NonStop AutoTMF software runtime DLL as a DLL of the user's library or DLL.

Changing the User Library with Prepare

NonStop AutoTMF software usually assumes that the application is fully installed and that programs and user libraries are properly configured before the preparation begins. If, however, you need to change the programs to refer to a different user library, you may do this as part of the NonStop AutoTMF preparation process.

To force PREPARE to change a library in an object file, use the LIBRARY option. The specified library must contain the NonStop AutoTMF runtime library. The LIBRARY option can be used when you need to change the location of a user library (even if the object file is already prepared).

```
AutoTMF 1? prepare myobj.myprog,library newobj.myusrlib;
--- $DATA.MYOBJ.MYPROG preparation complete
--- AutoTMF runtime library $DATA.NEWOBJ.MYUSRLIB
```

Preserving the Modification Timestamp of Object Files

Use a licensed Escort CI to preserve the modification timestamp of programs when they are prepared. To enable this feature, do the following:

1. In binder, type the following commands:

```
ADD * FROM escort
SET LIKE escort
ALTER CALLABLE^SET^MODTIME, CALLABLE ON
BUILD escort!
```

2. At the TACL prompt, SQL compile escort:

```
SQLCOMP/ IN escort/ NOREGISTER ON
```

3. Log on as SUPER.SUPER, and license escort:

```
FUP LICENSE escort
```

If your site is not licensed to run NonStop SQL, contact HP to request a copy of the privileged ESCORT CI.

Preserving License Attribute when Preparing Privileged Programs

Preparing a privileged licensed program disables the license attribute, unless the user issuing the prepare command is super.super. A warning message is displayed if the license is revoked:

```
*Warning* The LICENSE for your-object was disabled.
* 1667  * SUPER.SUPER must issue a FUP LICENSE command to restore it.
*       * See the AutoTMF documentation for a method to automatically
*       * preserve the license when preparing programs.
```

An alternative to issuing the command as super.super is to use a licensed Escort CI. To create a licensed command interpreter, do the following:

1. In binder, type the following commands:

```
ADD * FROM escort
SET LIKE escort
ALTER CALLABLE^SET^LICENSE, CALLABLE ON
BUILD escort!
```

2. At the TACL prompt, SQL compile escort:

```
SQLCOMP/ IN escort/ NOREGISTER ON
```

3. Log on as SUPER.SUPER, and license escort:

```
FUP LICENSE escort
```

If your site is not licensed to run NonStop SQL and you are running a version of NonStop AutoTMF software that predates SPR T0581AAP, contact HP to request a copy of the privileged ESCORT CI.

Tips for Preparing Programs

If you test NonStop AutoTMF software on a newly installed application, or on an application duplicated from your test system or production system, you should fully test that application before preparing the object files.

Newly installed applications or duplicated applications often fail because of errors in the PATHWAY configuration, incorrect ASSIGNs and DEFINEs, bad user-library pointers, and similar considerations. If you eliminate these sources of error before starting the NonStop AutoTMF preparation process, subsequent errors are more likely caused by the preparation and not by the application's setup.

Also, you should prepare all programs that are part of the application, even if you know that some programs will not require automatic transactions. This configuration is simple to manage and causes fewer errors when the application is run. Prepared programs that do not require transactions to access audited files run like unprepared programs; they do not generate additional transactions.

After you have prepared the object files, retest the application before auditing any database files. At this time, you can experiment with the NonStop AutoTMF tracing facility. For example, if you are testing a PATHWAY server or a batch program, enter the following ESCORT CI command before the program starts:

```
AutoTMF 1? trace to trcsv.myserver program data.srvobj.myserver;
Trace index       0
Tracing           EXECution
Trace file        $DATA.TRCSV.MYSERVER, Entry Seq, EOF=0 (0.0% full)
Trace expiration  May 9 2007  12:45:29 (59 minutes remaining)
Trace owner       170,46 (USER.SALLY)
Tracing programs  $DATA.SRVOBJ.MYSERVER
```

Once the program has run, examine the trace file.

Diagnosing Preparation Errors

The INFO PROGRAM command displays the preparation status of each object file:

- Unprepared Calls – Program calls to procedures that are normally intercepted by NonStop AutoTMF software are not retargeted. Either the program was not prepared, or was prepared using an earlier release of NonStop AutoTMF software. In either case, issue the PREPARE again.
- No User Library – The program has no user library. Either the program was not prepared or the user library was removed. In either case, issue the PREPARE again.
- Invalid User Library – The user library does not contain the NonStop AutoTMF runtime library procedures. Follow the procedures below for Preparing Programs that Have a User Library.
- Intercept Conflict – The program has a procedure that has the same name as an NonStop AutoTMF intercept procedure. NonStop AutoTMF procedure interception would not be reliable if the program were prepared.
- External Conflict – The program makes external calls to both prepared names and unprepared versions of the same procedure name; this situation would lead to duplicate externals (see next item) if the program were prepared.

  External conflicts occur when prepared and unprepared objects have been bound together; try to avoid this situation; however, if you need to bind prepared and unprepared objects, use the ALLOWDUPLICATES option (causing duplicate externals), then rebind the object file.
- Duplicate Externals – The program has duplicate names in the external procedure list, resulting in unresolved externals when the program is run, probably leading to program failure. Re-bind the object file.

  To avoid duplicate externals, build programs using only unprepared object files, then prepare them.
- Library Conflict – The program is not prepared but is configured to use a library that contains the AutoTMF runtime library, causing an inconsistency between the prepared state of the program and the prepared state of the program's user library. Issue the PREPARE command to prepare the program.

Note that access errors may prevent preparation of an object file; both read and write access are required.

Non-executable object files are not prepared unless they are to be bound with the NonStop AutoTMF runtime and used as a user library. Consult Preparing Programs that Have a User Library below. In most cases, you do not need to prepare separately compiled modules; you only need to prepare complete programs.

4

Configuring Automatic Transaction Processing

Configuration Commands

You can configure NonStop AutoTMF software features using commands to set global options and to specify file names and program names that use NonStop AutoTMF software options. See ALTER GLOBAL on page 6-34, ADD ATMFFILESET on page 6-6, and ADD ATMFPROGRAMS on page 6-12 for complete use and syntax descriptions of these commands. The ADD commands have corresponding INFO, ALTER and DELETE commands.

In unusual situations, you can also configure many of the same features by using DEFINES; however, configuring DEFINES consistently in a production environment is difficult and prone to error. See Appendix B, Special DEFINEs.

Configuring File Sets

When you configure ATMFFILESET and ATMFPROGRAMS, you specify file sets or patterns, such as $DATA*.FIRM*.LOG*. Because a file name may be configured by more than one pattern, an unambiguous search order of patterns has been defined. When an audited file is opened or program is started, the name is compared to the configured file sets. The first successful match concludes the search and determines the NonStop AutoTMF configuration options.

For example, suppose the following two commands were used to configure ATMFFILESET:

```
ADD ATMFFILESET $*.FIRM01.LOG* CREATEAUDIT;
ADD ATMFFILESET $DATA.*.LOGFILE SEPARATETX;
```

The search order causes the file name to be compared to $DATA.*.LOGFILE before $*.FIRM01.LOG*. If the file $DATA.FIRM01.LOGFILE is opened, the SEPARATETX option is applied, not CREATEAUDIT.

File Set Search Order

The search order for ATMFFILESET and ATMFPROGRAMS file sets places more specific file sets before more general file sets.

1. No wild-card in entire file name ($DATA.FIRM01.PAYROLL)
2. Wild-card only in file part of file name ($DATA.FIRM01.*)

3. Wild-card in subvolume (but not volume) part of file name ($DATA.FIRM*.PAY*)
4. Wild-card in volume part of file name ($D*.FIRM01.PAY*)

Wild-card names in the same part are also ordered by the location of the first wild-card character. For example, PAY* would precede P*.

To display the file sets in the defined search order for the current configuration, use the INFO ATMFFILESET and INFO ATMFPROGRAMS command. To show the current NonStop AutoTMF configuration for any existing file, use the detailed display of the FILEINFO command, for example:

```
AutoTMF 7? fileinfo $DATA.FIRM01.LOGFILE, detail;

$DATA.FIRM01.LOGFILE
    Enscribe
    Type U
    Ext( 2 Pages, 2 Pages, Maxextents 16 )
    Buffersize 4096
    Audit (AutoTMF: SEPARATETX)
    Buffered
    Owner 100,100 (DEV.GEORGE)
    Security (RWEP): NUNU
    Data Modif: 21Sep00 10:02
    Creation Date: 21Sep00 10:02
    Last Open: 21Sep00 10:03
    EOF: 0 (0.0% Used)
    Extents Allocated: 0
```

Command Options

In most cases, the NonStop AutoTMF features described in the following subsections can be configured by using ALTER GLOBAL, ADD ATMFFILESET, and ADD ATMFPROGRAMS.

Global parameters values are the default for all file and program attributes. Setting a program attribute overrides the default (global) value and setting a file attribute overrides the default (global) value and the program attribute.

Table 4-1. Configuration Command Options: Settings and Keywords

Feature	Global Settings	File Set settings	Program settings
	ALTER GLOBAL	ADD ATMFFILESET	ADD ATMFPROGRAMS
Automatic Transactions	AUTOTMF	NOTX	NOTX
Common Transaction	ATMFCOMMONTX	COMMONTX	COMMONTX
Separate Transactions	Not Applicable	SEPARATETX	SEPARATETX
Transaction Creation	Not Applicable	Not Applicable	Not Applicable

Table 4-1. Configuration Command Options: Settings and Keywords

Feature	Global Settings	File Set settings	Program settings
	ALTER GLOBAL	ADD ATMFFILESET	ADD ATMFPROGRAMS
Transaction Commit	ATMFMAXTIME ATMFMAXUPDATES	MAXTIME MAXUPDATES (For separate tx)	MAXTIME MAXUPDATES
Transaction Isolation	ATMFISOLATION	Not Applicable	WEAK NORMAL STRONG
Nowait Transactions	ATMFNOWAIT	NOWAIT (For separate tx)	NOWAIT
Forced Transaction Commit	ATMFAUTOCOMMIT	AUTOCOMMIT	AUTOCOMMIT
	ATMFNOWARNLONGTX	NOWARNLONGTX (For separate tx)	NOWARNLONGTX
Setting Transaction Time Out	ATMFTXTIMEOUT	TXTIMEOUT (For separate tx)	TXTIIMEOUT
Audited Attribute Hiding	Not Applicable	HIDEAUDIT	Not Applicable
Audited File Creation	Not Applicable	CREATEAUDIT	Not Applicable
Audited File Renaming	ATMFAUDITRENAME	Not Applicable	Not Applicable
Null Record Handling (Entry-Sequenced Files)	ATMFSKIPNULLRECS	SKIPNULLRECS READNULLRECS	Not Applicable
UNLOCKFILE Optimization	ATMFOPTIMIZEUNLOCKS	Not Applicable	OPTIMIZEUNLOCKS
Reading Through Locks	AMTFREADTHRULOCKS	READTHRULOCKS	READTHRULOCKS
Requiring Files to Be Audited	ATMFABENDNOAUDIT	Any ATMFFILE parameter including NOTX	Not Applicable
Record-Level Transactions	Not Applicable	RECORDTX	RECORDTX
Suppressing Inherited Transactions	Not Applicable	Not Applicable	SUPPRESSINHERITED TX

Table 4-1. Configuration Command Options: Settings and Keywords

Feature	Global Settings	File Set settings	Program settings
	ALTER GLOBAL	ADD ATMFFILESET	ADD ATMFPROGRAMS
Unstructured Access	Not Applicable	Not Applicable	Not Applicable
Large-Transfer Writing	Not Applicable	Not Applicable	Not Applicable
Transaction File (TFILE)	ATMFMAXTX	Not Applicable	MAXTX
Handling TMF Environmental Errors	ATMFSTOPONTMFERR	Not Applicable	STOPONTMFERR
Changing Nowait IO to Waited IO	Not Applicable	WAITEDIO	WAITEDIO

Automatic Transactions

NonStop AutoTMF software provides transactions when a program is not programmed to provide transactions. As needed, NonStop AutoTMF software starts and commits automatic transactions. Nontransactional programs are unaware that automatic transactions are being created and committed.

While automatic transactions are being created and committed, a process may have its own transactions, through calls to BEGINTRANSACTION or by inheriting transactions through messages on $RECEIVE. Such transactions are called process transactions. The process can manage its own transactions, while its automatic transactions are managed transparently by NonStop AutoTMF software.

Automatic transactions are created on demand, that is, only when the process accesses an audited file in a way that requires a transaction.

Automatic transactions are used only for database operations within a single process; automatic transactions are never exported to server processes.

Automatic transactions are always committed; they are never aborted.

Common Transaction

The common transaction is the default type of automatic transaction. The common transaction is used to access multiple audited files within the same transaction. At any time, a process has at most one common transaction.

Nontransactional applications do not require configuration of common transactions. You may configure common transactions to force the use of automatic transactions for specific files, even if processes have their own transactions. This capacity is useful

when a transactional application has some previously unaudited files that are to be audited using automatic transactions.

You may also use the ALTER GLOBAL ATMFCOMMONTX command to disable automatic transactions for all files that are not explicitly configured for those transactions.

Note. Do not configure files that were previously audited as COMMONTX, since configuring such files can cause locking problems.

Separate Transactions

Separate transactions are used to access a single file open (indicated by a unique file number) within its own transaction.

Configure separate transactions to:

- request automatic transactions for specific files, even if processes have their own transactions. (Consider configuring common transactions instead.)
- avoid long-running transactions in programs that have unusual locking behavior.
- force a single-transaction-per-update. Note that you also must specify MAXUPDATES 1 to force single-transaction-per-update. Each file configured through SEPARATETX and MAXUPDATES 1 uses its own separate transaction that is committed immediately after an update.

Other options you can configure for a separate transaction are NOWAIT, AUTOCOMMIT and MAXTIME.

A process may have both common and separate automatic transactions; all automatic transactions in a process are managed independently.

Note. Do not configure files that were previously audited (the application programs already access them under their own transaction) as SEPARATETX. This can lead to locking problems.

Transaction Creation

Automatic transactions are created on demand, whenever a process accesses an audited file and the process state and the NonStop AutoTMF configuration indicates that the access requires an automatic transaction.

Typically, an automatic transaction is created if a process accesses an audited file for a locking operation or an update operation, and that process does not have its own transaction. You can configure certain files for automatic transactions, either common or separate, regardless of the process transaction. You can also configure files that should not be accessed through an automatic transaction.

The first transactional access on a file OPEN determines if operations on that OPEN should use automatic transactions. Automatic transactions are used consistently, even if the program handles transactions inconsistently.

Transaction Commit

To commit an automatic transaction, NonStop AutoTMF software analyzes all file-access operations. File locking, record locking, and update operations are the primary factors that determine when automatic transactions may be committed.

Unaudited file-locking behavior and record-locking cannot be replicated when performing operations on audited files, but NonStop AutoTMF software ensures that normal, unaudited, file-locking behavior and record-locking behavior are observed.

NonStop AutoTMF software ensures that the database does not become inconsistent by preventing the premature unlocking of files and records. When the application has logically unlocked records and files, NonStop AutoTMF software can safely commit automatic transactions.

Automatic transactions are committed by a proprietary set of processing rules, driven by the following activities and events:

- Process operations that lock, update, and unlock files and records; such information determines both the lock protocols that need to be preserved and the actual lock state of open files.
- Process operations that signal a requirement or an opportunity to commit an automatic transaction. For example, when a server process replies to a requestor: the server has performed database operations for one request and will await a new request; any outstanding automatic transaction should be committed at this point.
- Process operations, such as termination, that would cause automatic transactions to be aborted.
- Process operations that may require a commit to isolate automatic transactions from external effects, as described in Transaction Isolation on page 4-7
- Time that an automatic transaction has been alive; you can configure the maximum transaction lifetime to a reasonable interval that is consistent with good performance
- Amount of update activity performed under an automatic transaction; you can configure the maximum number of updates in a transaction to keep the transaction overhead to a reasonable fraction of the total processing cost.

  Transactions that are used for a large number of updates accumulate locks, cause other resource problems, and have a larger impact if aborted unilaterally.

Transactions tend to increase the time during which locks are held and this increased time may result in increased lock contention. This phenomenon is common in all transaction-based database systems. Lock contention can be monitored by using the NonStop AutoTMF LISTLOCKS command.

Excessive lock contention can be reduced through proper NonStop AutoTMF configuration, but changes in the configuration may cause more frequent transactions and a corresponding increase in processing requirements.

Transaction Isolation

Transaction isolation influences the duration of automatic transactions to reduce the effect of automatic transactions outside the process that started them. There are three levels of isolation:

- Weak isolation commits active automatic transactions whenever a server process waits for or replies to a request on $RECEIVE. The default isolation mode is weak isolation, which is sufficient for most applications and provides the best performance; you should use weak isolation unless problems are clearly identified.
- Normal isolation also commits active automatic transactions whenever a process sends a request to a server process; normal isolation is required in applications where both requester and servers lock or update the same records.

  If processes coordinate access to the same database records (for example, one process inserts a database record, then sends a request to a server process to update the record), a record deadlock may occur.

  Coordinated access in an unaudited database requires a process to unlock a record before sending a server request to access that record. With automatic transactions, however, the record lock is held until the transaction is committed and the server may be blocked by the locked record.
- Strong isolation also commits active automatic transactions whenever a process sends a message to a device or process, or before the process updates an unaudited file; strong isolation is required only if it is unacceptable to report an update that might (with low probability) be later undone by a unilateral abort.

Note that regardless of the isolation setting, automatic transactions are not committed if the application program is still holding locks because releasing locks would violate the unaudited lock protocol and change the program’s behavior.

Nowait Transactions

To improve the performance of processes using automatic transactions, you can configure nowait transactions. In that case, NonStop AutoTMF software issues the ENDTRANSACTION call but does not wait for the transaction to complete. A nowait commit allows the process to continue with other work while the transaction commits in parallel.

For example, when server processes reply after making database updates, automatic transactions are committed. If you do not require servers to wait for commit processing, you can improve interactive response time, usually by a few tenths of a second.

Once a nowait transaction commit has begun, only a system-level failure can cause the transaction to abort. Even if the process terminates or is stopped externally, a nowait transaction commits successfully.

The only drawback to using nowait transactions is that the process continues to perform work without knowing if a unilateral abort has occurred. A server process replies to a requestor, but the database update operations the server process has performed may be backed out by a system-level failure.

Batch processes that have multiple, separate, automatic transactions may have better performance because the multiple transactions complete in parallel. The overall performance, however, depends on the number of updates performed in each transaction.

Each time an automatic transaction begins, any previous nowait transaction for the same entity must be completed. Thus, a batch program that uses only the common transaction may not benefit from nowait transactions, because the batch program needs a new transaction as soon as the previous transaction is committed.

NonStop AutoTMF software notifies you if any automatic transaction (nowait or waited) aborts by immediately abending the process with an explanatory message to the process home terminal and to the EMS log.

Audited Attribute Hiding

Most legacy programs are unaware of auditing and simply access the database as if the database were unaudited. Some programs, however, check the database and terminate (or otherwise malfunction) if audited files are found. If you configure HIDEAUDIT for a file, the following procedures will report that the file is not audited:

- FILEINFO
- FILERECINFO
- FILE_GETINFO_
- FILE_GETINFOBYNAME_
- FILE_GETINFOLIST_ (first item only)
- FILE_GETINFOLISTBYNAME_ (first item only)

Due to the complexity of locating a result item in a multi-item list request, the audit attribute is hidden only if the first item number is for the audit attribute (66).

Audited File Creation

Some programs create files, but the files are not audited. NonStop AutoTMF software allows you to configure file names to be created automatically as audited.

Note. The NonStop RDF product replicates the creation of an audited file on the backup node; the NonStop RDF product does not create a file on the backup node if a file is created as unaudited and is later altered to be audited.

Audited File Renaming

Attempts to rename a file while the file is audited are rejected by the file system with an error. Each audit record contains the name of the disk file being updated, so renaming a file can make backout operations fail.

NonStop AutoTMF software can rename audited files by performing a complex series of open, close, and alter operations. A file that is opened by other processes cannot be renamed.

Note. The NonStop RDF product does not replicate the rename of an audited file on the backup node because the rename operation is not audited.

Null Record Handling (Entry-Sequenced Files)

Records can be inserted into an entry-sequenced file but they cannot be updated.

If the file is not audited, all inserted records are written to the file. Programs expect to find a valid and complete record when reading the file.

If the file is audited and if a transaction is aborted, any non-committed record is removed from the file and a null record (a zero-length record) is inserted in the non-committed record's place.

In many cases, a null record will cause problems for a non TMF-aware legacy program because the program is not expecting to read zero-length records.

NonStop AutoTMF software can hide the existence of null records in an audited entry-sequenced file by discarding the records instead of returning them to the programs.

Reading Through Locks

Normally a READ or READUPDATE operation requires that the record being accessed is not currently locked. However, since the operation does not lock the record, nothing prevents concurrent access to the same record by multiple processes.

NonStop AutoTMF software does not provide automatic transactions for non-locking operations like READ. Normally, no transaction is required, but if a process performs a READLOCK followed by READ operation on the same record, an error or deadlock may occur.

To prevent errors and deadlocks, NonStop AutoTMF software automatically configures each file open for read-through-locks mode. This mode has no effect on locking operations such as READLOCK, only on non-locking access such as READ. You may disable this configuration by altering the ATMFREADTHRULOCK global or by specifying NO READTHRULOCK for a file set or program. If, however, a process attempts a READ or READUPDATE on a file that has an active automatic transaction, READTHRULOCK will always be set.

Requiring Files to Be Audited

The NonStop AutoTMF software manages access to audited files in a manner transparent to application programs. If the audited attribute is removed from any of the audited files it is managing, the application will continue to operate without any problem because NonStop AutoTMF software does not attempt to generate transactions for un-audited files.

In some environments, the accidental or deliberate disabling of audit on such files is considered a critical event.

You can enforce the requirement that a file be audited:

1. First set the global parameter ATMFABENDNOAUDIT to ON.
2. Then, configure the files for which auditing is critical with one of the following ATMFFILE options: COMMONTX, SEPARATETX, RECORDTX, or NOTX.

When any of the configured files is opened by a program, NonStop AutoTMF software checks the audit flag. If the file is not audited, the NonStop AutoTMF runtime abends the program and sends a message to the EMS log, thus preventing any update to occur.

Handling TMF Environmental Errors

By default, if a program encounters a TMF environmental error, NonStop AutoTMF software aborts the program because the program cannot continue processing without compromising data integrity. This has the side effect of producing large numbers of saveabend files at a time when the system is under stress.

You can configure NonStop AutoTMF software to stop, rather than abort, a program that encounters TMF errors. This eliminates the proliferation of saveabend files caused by TMF problems.

If NonStop AutoTMF software aborts or stops a program when a TMF error occurs, it sends a message to the EMS log describing the failure and displaying the TMF error that was encountered.

STOPONTMFERR applies to the following TMF errors:

The TMF errors that are affected by this parameter are:

- FETMFNOTRUNNING (82)
- FETMFNOTCONFIGURED (84)
- FEDEVICEDOWNFORTMF (85)
- FEBEGINTRDISABLED (86)
- FETRANSABRTOWNERDIED (90)
- FENODEVBUFSPACE (89)

- FETRANSABRTOWNERDIED (90)
- FETRANSABRTBADDBID (91)
- FETRANSABRTNETDOWN (92)
- FETRANSABRTAUDOVFL (93)
- FETRANSABRTOPRCMD (94)
- FETRANSABRTDISCTKOVR (95)
- FETRANSABRTTIMEOUT (96)
- FEABORTEDTRANSID (97)
- FENOMORETCBS (98)

Suppressing Inherited Transactions

A non TMF-aware server may inherit a transaction from a TMF-aware requester when reading $RECEIVE. The transaction has no effect on the program. However, if NonStop AutoTMF software is configured, the inherited transaction becomes visible to the program.

When NonStop AutoTMF software detects an inherited transaction, NonStop AutoTMF software assumes that the process will use that transaction to access audited files.

Suppressing inherited transactions insures that NonStop AutoTMF software generates automatic transactions for all audited file accesses, emulating the behavior of a non TMF-aware process.

UNLOCKFILE Optimization

Non TMF-aware programs sometimes issue a blanket call to UNLOCKFILE to release all locks rather than managing locks individually. In high-volume environments, the cost of UNLOCKFILE can be noticeable. When files are audited, the transaction commit unlocks records that participate in the transaction, making the UNLOCKFILE redundant.

NonStop AutoTMF software can be configured to eliminate the call to UNLOCKFILE. Since the program has released all locks on the file by issuing the UNLOCKFILE, NonStop AutoTMF software attempts to commit the transaction, following the usual rules for committing an automatic transaction. If a lock is held on another file participating in the transaction, the commit will occur when this lock is released by the program.

Note that this optimization can only be enabled for files that are also enabled for automatic transactions.

Record-Level Transactions

Certain programs use a complex record-locking and record-unlocking sequence such that at least one record is locked for long periods. NonStop AutoTMF software will not commit an automatic transaction until a program releases all locks obtained under the transaction. If the processing contains no point where all locks are released, the automatic transaction is never committed.

The resulting long-running transaction can cause lock contention, deadlocks, and, eventually, a unilateral abort when the TMF AutoAbort time limit is reached. Uncommitted updates could be lost. See the Unilateral Aborts discussion below.

In such cases, you can configure NonStop AutoTMF software to generate one automatic transaction for each record lock and to commit each transaction as soon as the record is unlocked. This approach increases the transactional activity but prevents the occurrence of long-running transactions without changing the program's locking behavior.

This feature is enabled by specifying the RECORDTX attribute on both the file on which the locks are held and the accessing program. If RECORDTX is specified on either the file or the program, but not on both, RECORDTX is ignored.

To implement RECORDTX, NonStop AutoTMF software maintains a table of records locked and the associated transaction for each lock. If the same record is locked multiple times, the same transaction is used for each lock until the lock is released and the transaction is committed.

Exceptions

Certain sequences of operations do not lend themselves to the generation of automatic transactions for each record lock. In these cases, the RECORDTX attribute is not supported and cannot be used:

- If a program issues READLOCK (In COBOL, READ NEXT WITH LOCK) to read records sequentially, NonStop AutoTMF software cannot predict the key of the next record that will be locked, so NonStop AutoTMF software cannot identify which transaction to use if the same record is locked more than once. As long as READLOCK is used to lock records that are not already locked, READLOCK may be used with RECORDTX.
- If a program does a LOCKFILE after locking two or more records, the file cannot be locked unless the records are unlocked first. Otherwise, the same record could be locked under more than one transaction, leading to internal deadlocks. If a LOCKFILE operation that has more than one record lock is requested, the LOCKFILE operation will be abended.
- If a program issues READUPDATELOCK with positioning by a unique alternate key, NonStop AutoTMF software cannot predict the primary key of the record being locked, so NonStop AutoTMF software cannot identify which transaction to use if the same record is locked more than once. As long as READUPDATELOCK is

used to lock records that are not already locked, READUPDATELOCK may be used with RECORDTX.

Considerations

As mentioned above, record-level transactions increase transactional activity and processing complexity, and they are useful only in very specific situations. Users should configure record-level transactions when a careful analysis of a program's locking behavior leaves no other option or after being instructed to do so by product support.

Changing Nowait IO to Waited IO

To commit automatic transactions, AutoTMF must find a point where no records are locked by the application program. Most programs have such points, but others have complex algorithms that lock records on some files before unlocking records on other files, leaving no commit opportunity. The configuration options SEPARATETX and, rarely, RECORDTX can be used in most cases to provide the necessary commit points.

However, a program may still be too complex to be managed using these methods. The WAITEDIO feature handles these rare cases, where a program opens a file multiple times for NOWAIT processing, and performs concurrent I/O to the same file. If the I/Os are performed under a common transaction, an I/O may be in progress at all times, preventing the transaction from being committed. If the I/Os are performed under separate transactions, operations that need to lock the same record on different opens may cause an internal deadlock.

If the WAITEDIO option is enabled for a NOWAIT file open, AutoTMF performs the following operations:

- Issues an AWAITIO immediately after the read, write, or update operation.
- Following the AWAITIO, AutoTMF commits any outstanding transaction using its usual algorithms, as directed by the configuration.
- Saves the values returned by AWAITIO, and returns control to the program as if the READ or WRITE operation were still in progress. After a READ, the data buffer contains the input data.

FILEINFO and FILE_GETINFO_ return the error code for the last READ or WRITE operation (instead of AWAITIO). Any error returned by the AutoTMF generated AWAITIO is not visible to the application until the application calls AWAITIO. When the program calls AWAITIO, AutoTMF returns the saved values and makes the operation error visible to the application.

Configuration

To enable waited I/O, configure both the ATMFFileset and ATMFProgram WAITEDIO attributes. A program configured using the WAITEDIO option uses Waited I/O only for audited files that are also configured for Waited I/O.

In addition to specifying WAITEDIO, use the WAITTIME option to specify the maximum number of seconds an operation will wait for record locks to be released.

Considerations:

- When AutoTMF performs AWAITIO, it does not know the timeout value that will be specified in the application call to AWAITIO. Configure the WAITTIME option on a file basis. To avoid perpetual waits, AutoTMF uses the default wait time of 64 seconds.
- AutoTMF does not return file system errors 27 and 28 when a program performs an I/O operation without calling AWAITIO on the previous operation. Instead, AutoTMF signals an assertion failure which commits all outstanding transactions and terminates the program.
- Error 26 occurs in the normal manner.
- When debugging a program, Inspect shows the completed operation state and error code as soon as the read or write operation has been executed.
- A FILEINFO, or a similar operation called between READ and AWAITIO can return updated key values that are normally available only after the call to AWAITIO.
- If executed between the READ/WRITE and AWAITIO, FILE_GETINFOLIST_ can return the wrong value for the last error code.
- Configuring WAITEDIO normally has a minor or negligible effect on performance. Overlapped NOWAIT I/O is very useful for non-audited files, where every update requires a physical disk operation. Since audited file updates are buffered, NOWAIT I/O is far less useful for audited files.

△ **Caution.** Not all programs that perform multiple NOWAIT operations will encounter this problem. Using this feature has some side effects. Users should not use Waited I/O except in consultation with product support.

Unstructured Access

The OPEN and FILE_OPEN_ procedures have an optional parameter to request unstructured access.

Unstructured access to a structured file (key-sequenced, entry-sequenced, or relative) allows a program to position to a relative byte address and to read or write a string of bytes at that location. Interpreting and updating the internal Enscribe block structures are the responsibility of the programmer.

Unstructured access to either unstructured files or structured files also permits large transfers to be used if the appropriate BULK I/O interfaces are used.

When a file is audited, some the following rules apply to the use of unstructured access:

- A structured audited file cannot be opened for write access. An attempt to perform such an open results in an error 80. TMF and the file system do not permit unstructured updates to an audited structured file. TMF auditing is based on the logical records of the file; unstructured access changes the physical structure of the file, so the changes cannot be properly reflected in the TMF audit.
- Unstructured read access to an audited file, structured or unstructured, is not restricted.
- Large transfer updates to unstructured audited files are now supported by the NonStop OS. The emulation of large transfers is described below in Large-Transfer Writing.

Some third-party programs perform unstructured read/write opens of a structured file. Careful analysis of third-party programs has revealed that in most cases, these programs did not attempt to update the file; read/write access was requested because read/write access is the default.

If an error 80 occurs on an unstructured open of an audited structured file, NonStop AutoTMF software will retry the open for read-only access. The retry allows some programs to run successfully.

Programs that do attempt to update the audited structured file will receive an error 49 on the write operation. No known method can make such programs work with NonStop AutoTMF software. If an error 49 occurs, either the file must be unaudited or the program changed to use conventional record updates.

Large-Transfer Writing

Large transfer reads are permitted on audited files without a transaction, so these reads are processed without emulation or automatic transactions.

Some applications use the file-system large-transfer facility to load large files quickly. If the application issues a SETMODE 141 call, the application can then perform 56 KB unstructured writes directly to the disk.

Until recently, such large-transfer write operations to an audited unstructured file were not permitted by the file system. To emulate large-transfer writes, NonStop AutoTMF software deblocked the large write into an equivalent number of 4KB writes. If the file was opened in nowait mode, all writes except the last were waited. This emulation offered a trade off between the efficiency of large transfers and the ability to audit and, thereby, replicate such files.

However, recent versions of the file system (G06.20 and later) support large transfers to audited unstructured files; if NonStop AutoTMF software is running on a system that

supports large transfers, it takes advantage of the feature to provide optimum performance.

NonStop AutoTMF software cannot emulate unstructured writes (either with large or small transfers) to audited structured files.

Transaction File (TFILE)

The *TMF Application Programmers Guide* describes the use of the TFILE ($TMP) to manage multiple transactions in a multi-threaded program. Because NonStop AutoTMF software also manages multiple transactions and transactions in parallel with the program transactions, NonStop AutoTMF software opens the TFILE. For this reason, the TFILE must be managed as a shared resource, even though the TFILE appears to the program as a dedicated resource.

NonStop AutoTMF software manages TFILE sharing effectively, using the TFILE to support parallel and nowait automatic transactions, and to manage process transactions. Your programs can use the TFILE in the conventional manner, but you are bound by TMF limits on the number concurrent transactions in a single process described below in Number of concurrent transactions.

Unilateral Aborts

When a program updates audited files within a transaction, an external factor may cause the transaction to be aborted and uncommitted updates to be rolled back. Such aborts are called unilateral, because they do not require the cooperation of the process that started the transaction. Factors causing unilateral aborts include:

- A process is terminated by another process.
- A server process that inherits a transaction process (or its CPU) fails before replying to the transactional request.
- A server process that inherits a transaction process performs an ABORTRANSACTION request.
- A CPU, a primary disk process, or a TMF component fails.
- A transaction exceeds the system AutoAbort time limit.
- The TMF operator aborts the transaction.

NonStop AutoTMF software prevents many sources of unilateral aborts and detects and deals with unilateral aborts that are not preventable. When a unilateral abort does occur, NonStop AutoTMF software achieves database consistency that is equivalent to unaudited database consistency by:

- Intercepting operations that would cause the process to terminate (such as ABEND or STOP) and committing any outstanding automatic transaction.

- Avoiding sending transactional requests to other processes so that those processes cannot cause unilateral aborts.
- Detecting unilateral aborts and abending the process. The program will not make database updates that assume that earlier updates were completed.
- Providing transaction isolation controls to prevent a unilateral abort from affecting external users or database consistency; see the preceding subsection Transaction Isolation on page 4-7.
- Providing an AUTOCOMMIT option that allows you to force all outstanding automatic transactions to be committed before the process abends, to avoid reaching the TMF AutoAbort time limit. See subsection Forced Transaction Commit below for details.

Unilateral aborts caused by hardware or system failures should be rare.

Reducing causes of unilateral aborts

To reduce other causes of unilateral aborts in production processing, perform these tasks:

- Configure TMF to handle the projected transaction load.
- If programs hold record locks and file locks for long periods, alter the TMF AutoAbort parameter to avoid aborting long-running automatic transactions. If TMF version 3.6 or later is running in your system, another option is to configure a timeout for automatic transactions. See Setting Transaction Time Out below.
- Avoid using the TACL STOP and the PATHWAY SHUTDOWN2 MODE IMMEDIATE commands unless you know that the programs being stopped do not have pending transactions. Use the STOP PROCESS command to externally stop server processes that are reading $RECEIVE

Forced Transaction Commit

As mentioned in the paragraph on Transaction Commit, NonStop AutoTMF software ensures that normal unaudited locking protocols are observed in order to preserve the integrity of the program logic. NonStop AutoTMF software assumes that programs are locking and unlocking records as needed to process a business transaction or unit of work. Automatic transactions are therefore not committed if a program is holding a lock on a record or a file.

The case where a program does not conform to the unaudited locking protocol presents a problem. If a program fails to unlock a record or a file, the program prevents NonStop AutoTMF software from committing automatic transactions. To deal with such situations, AutoTMF issues a warning in the EMS log to notify the operator of the long running transaction when a transaction runs longer than 5 minutes. The warning is repeated every 5 minutes, until the transaction is committed. (You can suppress these warnings by setting the file or program attributes NOWARNLONGTX or the global parameter ATMFNOWARNLONGTX)

Eventually, a long running transaction can reach the TMF AutoAbort limit, at which point the transaction is unilaterally aborted and uncommitted updates are rolled back, causing potentially irrecoverable data loss.

While forcing a commit before a program has released locks is tantamount to altering the program's logic and would leave the program in an uncertain state, failing to commit the transactions before reaching the unilateral abort is not an option.

To prevent the loss of data in cases where the program's behavior causes a unilateral abort, AutoTMF uses a configurable AUTOCOMMIT timer. The AutoTMF runtime monitors automatic transactions to determine if any have been active for more than the AUTOCOMMIT time limit.

The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If there are outstanding automatic transactions when the AUTOCOMMIT time limit is reached, the NonStop AutoTMF runtime takes the following actions:

1. Forces the commit of all automatic transactions,
2. Issues an EMS message,
3. Terminates the process with an abend.

The program has to be restarted but no data is lost.

To use the AUTOCOMMIT feature effectively, the time limit should be set to be lower than the TMF AutoAbort time limit by at least 5 minutes. The default value for AUTOCOMMIT is 115 minutes, based on the default TMF AUTOABORT time limit of 120 minutes. If you set the TMF AUTOABORT limit to a different value, you must set the AutoTMF AUTOCOMMIT accordingly (AUTOABORT - 5 minutes).

Setting Transaction Time Out

As mentioned in Reducing causes of unilateral aborts above, some programs might need to hold locks for a longer period than is permitted by the TMF AutoAbort setting. An alternative to increasing the AutoAbort time limit for all transactions is configuring a time out value for automatic transaction, provided your system is running TMF version 3.6 or later.

To specify how long an automatic transaction can live, set the TXTIMEOUT file or program attribute or the ATMFTXTIMEOUT global parameter. Defining a TXTIMEOUT value effectively sets the TMF AutoAbort time limit, but only for a specific program, a specific file or for all automatic transactions.

TXTIMEOUT value can be set as follows:

- to a value between 5 and 515 minutes
- to NEVER, to ignore the TMF AutoAbort setting

Note that the AutoTMF AutoCommit time limit always applies. If you configure a TXTIMEOUT value that exceeds the AutoCommit time limit, adjust the AutoCommit accordingly.

Limitations

Program Logic

- A program must make one call to OPEN or to FILE_OPEN_ before the first call to any procedure that requires an open file number. Such procedures include READ, WRITE, KEYPOSITION, CONTROL, SETMODE, and so on. These calls to OPEN or FILE_OPEN_ allow the NonStop AutoTMF runtime to allocate and initialize its data structures.

  This requirement applies only to the first file-system procedure call made by the program; this requirement is usually satisfied by the program's initialization routines that OPEN $RECEIVE to read the startup message. If a procedure that requires a file number is called prior to the first call to OPEN, the program will trap because the NonStop AutoTMF data structures have not been allocated.

- When calling I/O procedures such as READ and WRITE, a program must use a file number that was obtained from a call to OPEN or FILE_OPEN_. Programs that assume that the file system allocates file numbers serially (and uses hard coded file numbers) will not function correctly. Programs *can*, however, assume that $RECEIVE will have file number zero.

SQL tables

NonStop AutoTMF software does not currently support the generation of automatic transactions to access SQL tables.

Number of concurrent transactions

The NonStop AutoTMF software runtime opens the TFILE to manage multiple, concurrent, automatic transactions. The number of concurrent transactions allowed for each process depends on the version of the NonStop OS:

- One hundred (100) transactions on systems running versions of the NonStop OS prior to G06.20. It is also the NonStop AutoTMF software default.
- Up to 1000 transactions on systems running the NonStop OS version G06.20 or later. To allow more than the default 100 transactions, you must configure the ATMFMAXTX global parameter or set the program attribute MAXTX.

5 Usage Guidelines

This section describes typical uses and limitations of NonStop AutoTMF software, drawn from the experiences of customers.

Auditing Files

Once NonStop AutoTMF software is installed, the programs are prepared, and after NonStop AutoTMF configuration attributes have been specified, you can enable auditing of files. Enabling auditing can be performed by using the standard system utilities FUP and SQLCI, but the NonStop AutoTMF software command interpreter can also audit entire file sets with a single ALTER command.

The audit attribute may not be changed when a file is open, so you will need to stop applications that are using the file in order to change the audit attribute.

Non TMF-aware applications

Non TMF-aware applications do not access any audited file. By default:

- Auditing a file will cause NonStop AutoTMF software to generate automatic transactions to manage access to the file because there is no active transaction.
- Access to all audited files occurs under the common transaction.

However you can configure some files to be managed as a separate transaction.

TMF-aware applications

TMF-aware applications access audited files under their own transactions. These transactions are begun and ended or aborted by the application, not by NonStop AutoTMF software. The non-audited files that these applications also access can now be audited and managed by automatic transactions generated by NonStop AutoTMF software.

NonStop AutoTMF software does not start an automatic transaction if a process transaction is active. To generate automatic transactions to access newly audited files, explicitly configure these files by specifying the NonStop AutoTMF software file attribute COMMONTX or SEPARATETX.

In a mixed NonStop AutoTMF software and application controlled transaction environment, you must specify to NonStop AutoTMF software which files are managed by NonStop AutoTMF software and which files are managed by the application. To insure that NonStop AutoTMF software generates transactions for only for newly audited files, proceed as follows:

1. Create a list of all the files that are newly audited.
2. Set the global parameter ATMFCOMMONTX to OFF.
3. Configure the new audited files as COMMONTX or SEPARATETX.
   The choice of COMMONTX or SEPARATETX is determined by concurrency requirements. In most cases, configuring the newly audited files to be managed under one common automatic transaction (COMMONTX) will yield the desired results. Knowledge of the access patterns and thorough testing will determine if a particular file must be configured to be accessed under its own transaction (SEPARATETX).

Failure to configure the newly audited files properly can lead to deadlocks and other unpredictable errors.

Auditing Unstructured Files

Transactional applications rarely use audited unstructured files, but you may wish to audit them for replication and file recovery. NonStop AutoTMF software supports automatic transactions for accessing unstructured files. However, practical limitations restrict the use of audited unstructured files.

Object Files

Auditing object files is impractical. When an object file is used for the first time (or the first time after a cold load or a move or duplication), the operating system updates external references, thereby modifying the file. Because the operating system does not use transactions (automatic transactions cannot be generated in the operating system), the update and process execution will fail. Thus, an audited object file is unusable.

Edit Files

Although automatic transactions work with audited edit files, auditing edit files is impractical. Common editing operations on an edit file move the end-of-file (EOF) backward, but roll-forward processing of TMF (as well as HP NonStop RDF replication) does not move the EOF backward. Because edit files store an internal directory pointer at the last word before the EOF, an incorrect EOF makes the file unusable.

Unstructured Data Files

An unstructured data file can be audited. Moreover, if the end of file (EOF) is not moved backwards, the EOF can be rolled forward or replicated. The PURGEDATA

operation, which sets the EOF to zero, is audited and can be rolled forward or replicated properly.

Log files and SPOOLER files have been successfully replicated.

Preparing HP Utilities

Some HP products contain utility programs that create and update files, but do not provide the transactions required to access audited files. If you wish to audit and access these files using automatic transactions, you must prepare the programs.

Prepared HP products will continue to function properly, but known limitations may cause them to be ineffective in using automatic transactions. The following subsections describe common HP programs that you may want to use with NonStop AutoTMF software.

Note that when an HP utility you have prepared is updated, either by installing an SPR or a new PVU, you will need to re-prepare the utility. The utilities discussed in this section are the following:

FUP

When prepared, FUP works as usual on unaudited Enscribe files and has enhanced functionality for some commands that operate on audited Enscribe files.

A prepared FUP can copy records to an audited output file, create files that will be automatically audited, and rename audited files. However, a prepared FUP does not allow loading to an audited structured file.

FUP is a privileged license program. To retain FUP's license attribute, you must be logged on as super.super to prepare FUP, or have configured a licensed Escort command interpreter (CI) as described in Preserving License Attribute when Preparing Privileged Programs on page 3-10.

- If you have a configured a privileged Escort CI, or if you are logged on as SUPER.SUPER when running the Escort CI, simply prepare FUP (or a copy of FUP) and specify "!", to override the prohibition on preparing HP utilities.

For example:

```
> ESCORT PREPARE test.fup!;EXIT;
--- $DATA.TEST.FUP preparation complete
--- AutoTMF User Library $DATA.ESCORT.ESCRUNTM
```

- If you are not using a privileged Escort CI and are not logged on as SUPER.SUPER when issuing the PREPARE command, a warning notifies you that the prepared FUP is not licensed. You must then have the SUPER.SUPER user license your prepared copy of FUP.

  For example

  1. Prepare FUP specifying the “!”:

```
ESCORT PREPARE test.fup!;EXIT;
--- $DATA.TEST.FUP preparation complete
--- AutoTMF User Library $DATA.ESCORT.ESCRUNTM

*Warning* The LICENSE for $DATA.TEST.FUP was disabled.
* 1667  * SUPER.SUPER must issue a FUP LICENSE command to restore it.
*       * See the AutoTMF documentation for a method to automatically
*       * preserve the license when preparing programs.
```

  2. Have the SUPER.SUPER user license your prepared FUP:

```
>FUP LICENSE $data.test.fup
```

ENFORM

ENFORM uses transactions only to read an audited dictionary. ENFORM does not update files. ENFORM either reads existing files or creates and writes new files. The ENFORM LIST statement can read audited files; this statement reads through locks and does not require transactions.

The ENFORM FIND statement creates and writes a new unstructured file. Unfortunately, FIND files cannot be audited and replicated, because ENFORM processes a FIND in the following sequence. ENFORM:

1. Creates an unstructured temporary file that has a name like $DATA.#0001234.
2. Opens and writes information to $DATA.#0001234.
3. Renames $DATA.#0001234 to the desired name of the FIND file name.

If you configured NonStop AutoTMF software to create the temporary file as audited, the file creation and the writes would be replicated properly. The file rename, however, could not be done on an audited file. A rename operation on an audited file would require closing the temporary file, which would cause the disk process to purge the file immediately.

ENABLE

ENABLE is both a transactional and nontransactional program. ENABLE generates transactions if your ENABLE program specifies SET TMF ON.

ENABLE checks the file attributes of database files; if a file is audited and the program has not specified SET TMF ON, ENABLE generates an error message and refuses to access the file.

If you have existing ENABLE programs that do not specify TMF ON, you may configure HIDEAUDIT for files that will now be audited. Then, by preparing the ENABLGS server, you will be able to access audited files through an ENABLE application that assumes the files are not audited.

EDIT, TEDIT, VS

Prepared copies of these programs work with audited edit files; file updates are processed with automatic transactions. Replicating edit files though audit is not practical because of the backwards EOF problem described under Edit Files on page 5-2, above.

Spooler

The Spooler control files and collector files can be audited and replicated. After a NonStop RDF software takeover, the completed Spooler jobs can be accessed and printed on the remote system.

The components that need to be prepared are CSPOOL and SPOOL.

For a warm start, the files that need to be audited are the Spooler collector files and control files.

You should change the obey file for the Spooler cold start. The line that creates the spool data file should have AUDIT specified; alternatively, you could use a prepared FUP to create the audited files automatically.

You should also configure the Spooler control files to be created as audited. The typical CI command would be:

```
ADD ATMFFILESET $SYSTEM.SPL.SPL*, CREATEAUDIT;
```

After an abrupt primary system failure and a takeover on a remote system, the replicated spooler files may be in a state that requires rebuilding the control files. Special considerations apply when rebuilding the spooler control files. See paragraph Rebuilding Spooler Control Files below.

Rebuilding Spooler Control Files

A Spooler WARM START with the REBUILD option first creates a temporary file of format *$vol.#nnnnn* (the usual format for a temporary file), then renames the temporary

file to the spooler control file. Because the rebuilt control file is first created as a temporary file, the CREATEAUDIT attribute is not applied and the rebuilt file is not created as an audited file.

If for any reason a rebuild of the Spooler is necessary, proceed as follows:

1. Stop the Spooler.
2. Rebuild the control file by starting the Spooler with the REBUILD option.
3. Stop the Spooler once more.
4. Audit the new control file.
5. If the REBUILD is performed on the same system (if not required because of a takeover situation), move the control file to the backup system.
6. WARM START the Spooler.

SORT and SORTPROG

Do not prepare any FastSort components. The FastSort utilities use bulk file transfers to access the input, scratch, and output files, which improves performance. However, such transfers cannot be audited.

FTP

NonStop AutoTMF software supports FTP file transfers to audited files. To enable this feature, do the following:

1. In ESCORT CI, prepare the FTPSERV object file.
2. In FUP, license FTPSERV.
3. In ESCORT CI, configure the target files for the transfer to be created as audited files:

```
ADD ATMFFILESET fileset-pattern, CREATEAUDIT;
```

The transferred files can then be replicated using RDF or another replication method that uses the audit trail.

File Transfer Processing

- A process Exclusive OPEN of an audited, unstructured file is converted to a Shared OPEN and the file is locked with a LOCKFILE operation under an automatic transaction.
- On an audited, unstructured file, OPEN for "unstructured access" and SETMODE 141,1 operations are suppressed. The disk process supports large transfers to unaudited, unstructured files without requiring the OPEN option or the SETMODE. Both operations interfere with writing to an audited file.

- The LOCKFILE operation is performed under one automatic transaction. Writes to the file do not acquire record locks when the entire file is locked. All writes to the file are performed under a single transaction, which is committed when the file is closed. If the file is not closed before the TMF AutoAbort time limit, the program fails.
- Tracing shows the alteration of the open exclusion and the deblocking operations.

△ **Caution.** Do not configure SEPARATETX for an audited file that is opened with the exclusion mode Process Exclusive. Doing so causes the process to deadlock attempting to LOCKFILE the file twice under different transactions

Preparing Third-party Applications

Numerous popular applications from independent software vendors have been successfully converted to use audited files with NonStop AutoTMF software. The conversion has not required vendor source code or product support from the vendors. Contact NonStop AutoTMF software product support for technical notes about converting these applications.

Auditing Enscribe Queue Files

An Enscribe Queue File is a special type of key-sequenced file that can function as a queue. Processes can queue and dequeue records to and from a queue file.

AutoTMF offers basic support for audited queue file.

AutoTMF processes a READUPDATELOCK operation on a Queue File like an UNLOCKREC or a DELETE operation. Thus, the record is not considered locked, allowing an automatic transaction to be committed.

- Prior to executing the READUPDATELOCK operation, AutoTMF attempts to commit any outstanding automatic transaction for the Queue File. MAXUPDATES, MAXTIME, and isolation settings are ignored and the automatic transaction is committed, if there are no logical locks held by the program.
- If the READUPDATELOCK operation times out (error 162), AutoTMF proceeds as follows:

  1. Returns the error 162 to the program, signaling to the program that it should retry the READUPDATELOCK operation.
  2. Commits the current automatic transaction.
  3. Begins a new automatic transaction.

See Chapter 7 in the *Enscribe Programmer's Guide* for a full description of the functionality and usage of Queue Files.

NonStop AutoTMF and Process Pairs

If AutoTMF is used, no error 75 occurs on a takeover because AutoTMF creates new automatic transactions in the backup process to access audited files.

However, when a primary process fails, automatic transactions may or may not have been aborted, depending on the cause of the failure.

If the failure is due to a programming error, such as a trap or a call to STOP or ABEND, AutoTMF commits all outstanding transactions before the program stops.

If the failure is due to an environmental error, such as the loss of a CPU, then the active transactions are unilaterally aborted.

After the primary process failure, the backup process takes over at some point in the processing, without knowing if all the previous updates have been committed.

In theory this situation could be remedied by making AutoTMF commit all active transactions whenever the primary process does a CHECKPOINT to the backup process. However, to commit transactions at every checkpoint would require that no record locks or file locks be held by the program when it calls CHECKPOINT. If the program still held locks, AutoTMF would have to abend the process after committing the transaction.

How to Use AutoTMF with Process Pairs

In T0581AAJ, AutoTMF was enhanced to intercept the CHECKMONITOR procedure and insure that the backup process of a process pair always initializes the AutoTMF internal segment upon startup to insure the AutoTMF environment in the backup process is ready in the event of a takeover.

However, be aware that

- No reliable way exists to checkpoint transactions.
- In some cases, a CPU failure may cause the database updates to be rolled back.

When the backup process takes over, AutoTMF starts generating new automatic transactions without knowledge of the transactions that occurred in the failed primary process. The backup process behaves as if the work had been completed and does not reapply an update that was rolled back.

Recommendation

To limit the scope of the recovery required in the backup process, you should set the ATMFProgram attribute MAXUPDATES be set to 1.Setting MAXUPDATES to 1 forces AutoTMF to commit transactions immediately after an update. Then if a failure occurs, at most one update must be recovered by the backup process.

6

NonStop AutoTMF Software Commands

The command interpreter (CI) is the primary interface for installing, configuring, controlling and monitoring NonStop AutoTMF software.

This chapter describes the command interpreter and is organized as follows:

Running the Command Interpreter

Use this TACL RUN command to start the command interpreter:

```
[RUN] $SYSTEM.ESCORT.ESCORT[/run-options/] [monitor;] [cmd]
```

run-options

process options for any TACL RUN command.

monitor

If an alternate monitor is configured, specifies the name of the alternate monitor process. The monitor process name must be a local name, starting with a dollar sign (“$”).

cmd

specifies a command.

The CI sets the completion code when the CI stops. The completion code values are:

0: Normal completion

1: Warnings messages were issued during the session

2: Error messages were issued during the session

During a CI session, you can display the completion code by using the ENV command. Use the RESET command to restore the completion code to 0. The RESET command also restores all parameters displayed by the ENV command to their default values.

Command Syntax

The CI is a conversational-mode program. Most commands can be continued over many lines without a continuation character; therefore, use a semicolon (;) to terminate each command.

Some commands, such as FC and RUN, are single-line commands and are terminated by the end of line; such commands can be continued by placing an ampersand (&) at the end of the line.

Commands are free-format: spaces are not significant except within character strings and numbers. The commands are not case-sensitive.

Guardian defines can be used to specify command parameters:

- CLASS MAP defines to specify parameters that designate files or tables
- CLASS CATALOG defines to specify command parameters that designate subvolumes

Command Summary

The AutoTMF commands are shown in Table 6-1.

The monitor process also accepts commands from the CI; the commands are shown in Table 6-2. To enter a monitor command, precede the command with the keyword MONITOR.

Common programs in the $SYSTEM.SYSTEM (or SYSnn) subvolume can be run directly by typing the name of the program; these programs are shown in Table 6-3; refer to the RUN[D] command for the complete syntax. TACL macros cannot be run from the CI.

Table 6-1. NonStop AutoTMF Software Commands

Command Name	Description
ABEND MONITOR	Abends the monitor process.
ADD ATMFFILESET	Defines NonStop AutoTMF software configuration for a file set
ADD ATMFPROGRAMS	Defines NonStop AutoTMF software configuration for a program file set
ALTER ATMFFILESET	Alters an NonStop AutoTMF software configuration for a file set
ALTER ATMFPROGRAMS	Alters an NonStop AutoTMF software configuration for a program file set
ALTER FILE	Changes the file attributes of a file set
ALTER GLOBAL	Changes the value of a global value

Table 6-1. NonStop AutoTMF Software Commands (continued)

Command Name	Description
ALTER MAPDB	Changes the attributes of an NonStop AutoTMF software MapDB and monitor
ALTER MONITOR	Changes the attributes of an NonStop AutoTMF software monitor process
ALTER LOCAL	Changes the value of an NonStop AutoTMF software global value (local) for the current monitor
CALC	A basic four-function calculator with accuracy to four decimal places
COMMENT (or “--”)	Performs no action; useful to document an interactive session or OBEY file
COPY	Copies data from any input file to any output file
CPUS	Displays the CPU configuration of a node
CREATE MAPDB	Creates a database for NonStop AutoTMF software configuration
CREATE SYSDB	Creates the NonStop AutoTMF software system database
DEADLOCK	Creates a deadlock on a file
DELAY	Suspends the execution of the command interpreter
DELETE ATMFFILESET	Removes NonStop AutoTMF software configuration for a file set
DELETE ATMFPROGRAMS	Removes NonStop AutoTMF software configuration for a program file set
DEQUEUE	Dequeues entries from a queue file
DROP MAPDB	Deletes a mapping database
ENV	Displays current volume and other environmental settings
FACTOR	Calculates prime factors for a given numeric expression
FC and !	Re-runs a previous command
FILEINFO	Displays information for Enscribe files
FILES	Displays a simple list of all files
HISTORY	Displays a list of previous commands
INFO ATMFFILESET	Displays NonStop AutoTMF software configuration for a file set
INFO ATMFPROGRAMS	Displays NonStop AutoTMF software configuration for a program file set
INFO GLOBALS	Displays NonStop AutoTMF software global values
INFO MAPDB	Displays MapDB information in the system database
INFO LIBRARY	Displays (and optionally STOPs) all processes executing with a specified user library

Table 6-1. NonStop AutoTMF Software Commands (continued)

Command Name	Description
INFO LOCALS	Displays NonStop AutoTMF software global values (locals) for the current monitor
INFO PREPARE	Displays external procedure names altered by PREPARE
INFO PROGRAM	Displays the NonStop AutoTMF software preparation status of an object file (equivalent to PROGINFO)
LABELDISPLAY (LD)	Displays file label information for a SQL table or View
LISTFILEOPENS (LFO)	Displays file open information
LISTLOCKS (LL)	Displays locked records and keys for a file set
LOG	Output a copy of command input and output to a file
MODIFY AUTOTMF	Modifies the system name in file and program configuration
MODIFY GLOBALS	Modifies the system number in global parameters
MODIFY MAPDBS	Modifies the system name in Monitor and MapDB configuration
MONITOR	Sends a command to a monitor process
NSKFIXUP	Performs an object file fixup
OBEY	Runs a sequence of commands from a file
OPEN	Sets the CI session mapping database and monitor process
OUT	Directs all command output to a file
PREPARE	Alters an object file to use NonStop AutoTMF software
PROGINFO (PI)	Displays information about object files (equivalent to INFO PROGRAM)
PURGEDATA	Clears all data from a file
RESET	Resets all environmental settings to their starting values
RESET GLOBAL	Resets a global parameter to its default value
RESET LOCAL	Resets a local parameter to its default value
RUN[D]	Runs an external process during a ESCORT CI session
START MONITOR	Starts monitor processes
STATS	Examines an Enscribe structured file
STATUS TRACE	Request information about active traces
STOP MONITOR	Stops the monitor process
STOP PROCESS	Stops an application process allowing automatic transactions to commit.
TIME	Displays the current time and date
TRACE	Initiates a trace for a program or set of programs
UNPREPARE	Returns an object file to its original state

Table 6-1. NonStop AutoTMF Software Commands (continued)

Command Name	Description
UPDATE	Updates a file record
UPGRADE MAPDB	Creates additional MapDB tables when required
VOLUME	Sets the CI session volume and subvolume

Table 6-2. Monitor commands

Command Name	Description
LOG	Starts and stops monitor activity logging
STATUS	Obtains status of various objects
SECURITY	Controls which users are permitted to issue monitor commands
BACKUPCPU	Alters the backup cpu of the monitor process
SWITCH	Causes primary and backup processes to exchange roles

Table 6-3. Executable External Programs

AXCEL	BIND	C	COBOL85
DCOM	DDL	DSAP	ECOBOL
EDIT	ELD	ENABLE	ENFORM
ENMC	ENOFT	EPTAL	ERROR
FTP	FUP	INSPECT	MEASCOM
NLD	NMC	NMCOBOL	NOFT
OCA	PATHCOM	PERUSE	PTAL
SAFECOM	SPOOLCOM	SQLCI	SQLCOMP
TAL	TEDIT	TMFCOM	TNSVU
VPROC			

Command Descriptions

ABEND MONITOR

Causes the monitor to stop and produce a saveabend file. Use this command only when you are required to supply information about a monitor problem and are instructed to do so by product support.

```
ABEND MONITOR [ * | process-name | mapdb ];
```

*

(asterisk) stops all configured monitor processes.

process-name

specifies the name of the NonStop AutoTMF software monitor process to be stopped. The default is the current monitor for the session.

mapdb

specifies a subvolume of a MapDB.

ADD ATMFFILESET

Configures automatic transactions for selected files. Files are specified by patterns that may select a single file or all files that match the pattern. The configuration for a file set

can be altered by using the ALTER ATMFFILESET and deleted by using the DELETE ATMFFILESET commands.

```
ADD ATMFF[ILESET] file-set [ , attribute ] ...;

attribute is

{ AUTOCOMMIT n [ SEC[ONDS] | MIN[UTES] ]                    }
{                                                           }
{ COMMONTX                                                  }
{                                                           }
{ CREATEAUDIT                                               }
{                                                           }
{ HIDEAUDIT                                                 }
{                                                           }
{ MAXTIME 1-to-n-seconds                                    }
{                                                           }
{ MAXUPDATES 1-to-n                                         }
{                                                           }
{ NOTX                                                      }
{                                                           }
{ NOWAIT[TX]                                                }
{                                                           }
{ NOWARNLONGTX                                              }
{                                                           }
{ READNULLRECS                                              }
{                                                           }
{ READTHRULOCK                                              }
{                                                           }
{ RECORDTX                                                  }
{                                                           }
{ SEPARATETX                                                }
{                                                           }
{ SKIPNULLRECS                                              }
{                                                           }
{ TXTIMEOUT { value | NEVER | OFF }                         }
{                                                           }
{ WAITEDIO [, WAITTIME 1-to-n seconds]                      }
```

file-set

TACL-style file name pattern specifying a collection of files.

attribute

AUTOCOMMIT *n* [SEC[ONDS] | MIN[UTES]]

(*with SEPARATETX only*)

instructs NonStop AutoTMF software to commit all automatic transactions if the separate transaction for this file exceeds the specified time limit.

n is a number between 300 and 30000 but stored internally in minutes, so if specified in seconds some rounding of the value may occur. The default unit is minutes.

If *n* = 0, AUTOCOMMIT is disabled for this file set.

If a program fails to unlock a record and prevents NonStop AutoTMF software from committing automatic transactions, the non-committed updates are eventually lost because the TMF AUTOABORT timer will cause the abort of the long running transaction.

To prevent such data loss, AUTOCOMMIT should be set to a value that is less than the TMF AUTOABORT timer value, about 5 minutes or so.

The default AUTOCOMMIT value is 115 minutes, based on the default TMF AUTOABORT limit of 120 minutes.

When AUTOCOMMIT is specified, the runtime monitors automatic transactions to determine those that have been active for more than the selected time. The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If any automatic transaction has exceeded the AUTOCOMMIT time, the NonStop AutoTMF software runtime will take the following actions:

1. force the commit of all automatic transactions,
2. issue an EMS message,
3. terminate the process with an abend.

The program will have to be restarted but no data will be lost.

To set this value for specific programs, use command ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

AUTOCOMMIT overrides the ATMFAUTOCOMMIT global.

IF AUTOCOMMIT is specified, SEPARATETX must also be specified.

COMMONTX

configures files for access using automatic transactions. Use this option to manage file access under automatic transactions (even if the process has a transaction), for files that are not configured for SEPARATETX.

CREATEAUDIT

specify the audit attribute on a file that is created programmatically.

HIDEAUDIT

requests that the audit attribute on a file be concealed from the application program when the program inquires about file attributes using procedure calls such as FILEINFO, FILERECINFO, FILEGETINFO, and so on.

Be specific when defining the fileset. HIDEAUDIT hides the audit attribute for all files in the fileset, even if the audited file is not intended to be accessed under a NonStop AutoTMF software automatic transaction.

Note. If the program calls FILE_GETINFOLIST_ or FILE_GETINFOLISTBYNAME_, the audit attribute is only hidden if the check for audit is the first in the item list passed to the procedure.

MAXTIME *1-to-n-seconds*

(*with SEPARATETX only*)

specifies that a separate transaction should be committed after *n* seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

IF MAXTIME is specified, SEPARATETX must also be specified.

MAXUPDATES *1-to-n*

(*with SEPARATETX only*)

specifies that a separate transaction should be committed after *n* updates and inserts. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

IF MAXUPDATE is specified, SEPARATETX must also be specified.

NOTX

disables automatic transactions for audited file access.

NOWAIT[TX]

(*with SEPARATETX only*)

returns control to the application without waiting for the transaction to be fully committed. Separate transactions are committed in nowait mode. The status of a nowait transaction is checked by the runtime when the next transaction for the file is started or when the process terminates.

IF NOWAIT[TX] is specified, SEPARATETX must also be specified.

NOWARNLONGTX

(*with SEPARATETX only*)

suppresses the warning message that is normally displayed in the EMS log when an automatic transaction runs longer than 5 minutes. The warning is repeated every 5 minutes until the transaction is committed.

Users should exercise caution when suppressing the long transaction warning and do so only if the cause for the delayed commit is well understood.

IF NOWARNLONGTX is specified, SEPARATETX must also be specified

READNULLRECS

(*for entry-sequenced files only*)

reverses the effect of the SKIPNULLRECS option. See the description of SKIPNULLRECS below for details.

To instruct the runtime to read null records for all audited entry-sequenced files, set the global parameter ATMFSKIPNULLRECS to OFF. To request this option for a specific program, use the =_ESCORT_READ_NULL_RECS define.

If this option is specified, it overrides the value of the global parameter ATMFSKIPNULLRECS and of the =_ESCORT_SKIP_NULL_RECS define.

READNULLRECS is the default.

READTHRULOCKS

instructs NonStop AutoTMF software to enable read-thru-lock mode on audited files that use automatic transactions.

If this READTHRULOCKS is specified, the file attribute value overrides the global parameter ATMFREADTHRULOCK value.

To set this option for a specific program file set use ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

RECORDTX

generates an automatic transaction for each locked record and commits the transaction as soon as the record is unlocked.

RECORDTX is used when a file is accessed by programs in which record locking sequences do not allow NonStop AutoTMF software to commit automatic transactions (there is at least one lock held on a record in this file at any given time).

For RECORDTX to take effect, it must be configured both as a program attribute and as a file attribute, using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

The RECORDTX file attribute can be specified in conjunction with other attributes, such as COMMONTX or SEPARATETX. The other attributes are ignored if RECORDTX is specified for both the file and the program. If RECORDTX is specified for a program, but not the file, then the other options, such as COMMONTX or SEPARATETX, will be in effect.

For further details on the use of RECORDTX, see paragraph Record-Level Transactions on page 4-12.

SEPARATETX

configures files to be accessed by the programs using separate automatic transactions. Specify SEPARATETX when automatic transactions for a file are always to be used (even if the process has its own transaction) and the transaction should be managed separately from transactions for other files.

SKIPNULLRECS

(*for entry-sequenced files only*)

instructs the NonStop AutoTMF software runtime to ignore zero-length records when reading sequentially through an audited entry-sequenced file.

If a record is inserted into an audited entry-sequenced file and subsequently backed out because a transaction is aborted, a zero length record is left in the file where the record had been inserted. This cannot occur if the file is not audited and can cause problems for programs that are not expecting to encounter such records. SKIPNULLRECORDS shelters the application programs from these unexpected records.

Although automatic transactions are never aborted by NonStop AutoTMF software, unilateral aborts could leave such gaps in an entry-sequenced file.

By default null records are not skipped.

TXTIMEOUT { *value* | NEVER | OFF }

(*with SEPARATETX only*)

specifies how long automatic transactions may live, regardless of the TMF AutoAbort setting. This feature requires TMF version 3.6 or later.

- *value* is *n* [SEC[ONDS] | MIN[UTES]]
  value can be between 5 and 510 minutes and overrides the TMF AutoAbort timer.
- NEVER specifies that the TMF AutoAbort timer is not in effect.
- OFF specifies that the TMF AutoAbort is in effect.

If TXTIMEOUT is configured on a system that is not running TMF 3.6 or later, a warning message is issued, the configuration is updated and the transaction timeout takes effect when TMF 3.6 is installed.

Note. The AutoTMF AutoCommit timer always applies, regardless of the configured transaction timeout. If the desired transaction time out value exceeds the default AutoCommit timer value, the user should adjust the AutoCommit timer accordingly. See AutoCommit attribute description above on page 6-7 for details.

IF TXTIMEOUT is specified, SEPARATETX must also be specified

WAITEDIO [, WAITTIME *1-to-n-seconds*]

converts NOWAIT I/O operations to WAITED I/O operations.

If WAITEDIO is specified for a NOWAIT file open, AutoTMF automatically performs an AWAITIO operation immediately following the various forms of READ and WRITE. After the AWAITIO completes, AutoTMF commits any outstanding transactions using the usual algorithms as directed by the configuration.

AutoTMF saves the values returned by AWAITIO, and returns control to program as if the READ or WRITE operation were still in progress. For a READ, the data buffer contains the input data.

WAITTIME sets the maximum number of seconds an operation will wait for locks to be released. The value specified is from 1 to 4096 and rounded to the nearest power of 2 (1,2,4,8...). The default is 64 seconds.

For this option to take effect, specify it both as an attribute of a program and as an attribute of the file using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

For further details, see Changing Nowait IO to Waited IO on page 4-13.

ADD ATMFPROGRAMS

Configures automatic transactions for selected programs. Programs are specified by file pattern that may select a single program or all programs that match the pattern.

The configuration for a program file set can be altered with the ALTER ATMFPROGRAMS and deleted with the DELETE ATMFPROGRAMS commands.

```
ADD ATMFP[ROGRAMS] object-fileset [ , attribute ] ...;

attribute is

{ AUTOCOMMIT n [ SEC[ONDS] | MIN[UTES] ]}
{                                                   }
{ COMMONTX                                          }
{                                                   }
{ ISOLATION { WEAK | NORMAL | STRONG }              }
{                                                   }
{ MAXTIME 1-to-n-seconds                            }
{                                                   }
{ MAXTX 1-to-1000                                   }
{                                                   }
{ MAXUPDATES 1-to-n                                 }
{                                                   }
{ NOTX                                              }
{                                                   }
{ NOWAIT[TX]                                        }
{                                                   }
{ NOWARNLONGTX                                      }
{                                                   }
{ OPTIMIZEUNL[OCKS]                                 }
{                                                   }
{ READTHRULOCK                                      }
{                                                   }
{ RECORDTX                                          }
{                                                   }
{ SEPARATETX                                        }
{                                                   }
{ STOPONTMFERR                                      }
{                                                   }
{ SUPPRESSINHERITEDTX                               }
{                                                   }
{ SUPPRESSUSERTX                                    }
{                                                   }
{ TXTIMEOUT { value | NEVER | OFF }                 }
{                                                   }
{ WAITEDIO                                          }
```

object-fileset

is a TACL-style file name pattern specifying a collection of object files

attribute

AUTOCOMMIT *n* [SEC[ONDS] | MIN[UTES]

instructs NonStop AutoTMF software to commit all outstanding automatic transactions if any exceeds the specified time.

n is a number between 300 and 30000 but stored internally in minutes, so if specified in seconds some rounding of the value may occur. The default unit is minutes.

If *n* = 0, AUTOCOMMIT is disabled for this program set.

If a program fails to unlock a record and prevents NonStop AutoTMF software from committing automatic transactions, the non-committed updates are eventually lost because the TMF AUTOABORT timer will cause the abort of the long running transaction.

To prevent such data loss, AUTOCOMMIT should be set to a value that is less than the TMF AUTOABORT timer value, about 5 minutes or so.

The default AUTOCOMMIT value is 115 minutes, based on the default TMF AUTOABORT time limit of 120 minutes.

When AUTOCOMMIT is specified, the runtime monitors automatic transactions to determine those that have been active for more than the selected time. The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If any automatic transaction exceeds the AUTOCOMMIT time, the NonStop AutoTMF software runtime takes the following actions:

1. force the commit of all automatic transactions,
2. issue an EMS message,
3. terminate the process with an abend.

The program will have to be restarted but no data will be lost.

To set this value for a specific fileset, use command ADD ATMFFILESET or ALTER ATMFFILESET.

Specifying AUTOCOMMIT as a program attribute overrides the ATMFAUTOCOMMIT global setting.

COMMONTX

configures files for access using automatic transactions. Use this option to manage file access under automatic transactions (even if the process has a transaction), for files that are not configured for SEPARATETX.

ISOLATION { WEAK | NORMAL | STRONG }

level of transaction isolation for automatic transactions:

- WEAK isolation commits outstanding transactions whenever a process replies to a request or waits for a new request. This is the default.
- NORMAL isolation commits outstanding transactions whenever a process sends a request to another process.

- STRONG isolation commits outstanding transactions whenever the process does one of the following:
  - issues a READ or a WRITE to a device,
  - issues a WRITE to a non-audited disk file

ISOLATION overrides the value of the global parameter ATMFISOLATION.

MAXTIME *1-to-n-seconds*

specifies that transactions should be committed after *n* seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default is 16 seconds.

MAXTX *1-to-1000*

specifies the maximum number of concurrent transactions a program can process. MAXTX is rounded up to the next multiple of 100 when NonStop AutoTMF software opens the TFILE ($TMP).

The default is 100.

If an application program opens the TFILE with a transaction depth greater than 100, NonStop AutoTMF software also increases its current value of MAXTX to the value specified by the program, rounded up to the next multiple of 100.

An application program must open the TFILE before NonStop AutoTMF software opens the TFILE in order to have the maximum depth increased under program control. Once NonStop AutoTMF software has opened the TFILE, NonStop AutoTMF cannot close and reopen the TFILE with the desired depth. In such cases, configure MAXTX specifically if a value larger than 100 is desired.

MAXTX overrides the value set through the global parameter ATMFMAXTX.

Note. You can set MAXTX to a value greater than 100 only on NonStop servers that support a maximum of 1000 transactions (T9055AER and later)

MAXUPDATES *1-to-n*

specifies that transactions should be committed after *n* updates and inserts. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default is 32.

NOTX

disables automatic transaction processing in the program.

NOWAIT[TX]

returns control to the application without waiting for the transaction to be fully committed. Automatic transactions generated by the programs are committed in nowait mode. The status of a nowait transaction is checked by the runtime when the next transaction for the file is started or when the process terminates.

NOWARNLONGTX

suppresses the warning message that is normally displayed in the EMS log when an automatic transaction runs longer than 5 minutes. The warning is repeated every 5 minutes until the transaction is committed.

Users should exercise caution when suppressing the long transaction warning and do so only if the cause for the delayed commit is well understood.

To suppress the warning for all programs, set the global parameter ATMFNOWARNLONGTX to ON.

OPTIMIZEUNL[OCKS]

specifies that UNLOCKFILE operations performed on audited files by the program fileset are suppressed. NonStop AutoTMF software considers all locks on the file released and attempts to commit the transaction, subject to the usual protocol for committing automatic transactions.

This optimization only applies to files that are enabled for automatic transactions.

UNLOCKFILE optimization can also be enabled for a program by specifying the class map define =_ESCORT_OPTIMIZEUNLOCKS with a dummy file.

To enable the optimization for all programs, set the global parameter ATMFOPTIMIZEUNL[OCKS] to ON.

READTHRULOCKS

instructs NonStop AutoTMF software to enable read-thru-lock mode on audited files that use automatic transactions.

If READTHRULOCKS is specified, the program attribute overrides the global ATMFREADTHRULOCK. setting

To set this option for a specific file set use the ADD ATMFFILESET or ALTER ATMFFILESET command.

RECORDTX

generates an automatic transaction for each locked record and commits the transaction as soon as the record is unlocked.

RECORDTX is used when a file is accessed by programs in which record locking sequences do not allow NonStop AutoTMF software to commit automatic transactions (there is at least one lock held on a record in this file at any given time).

For RECORDTX to take effect, it must be configured both as a program attribute and as a file attribute using ADD ATMFFILESET or ALTER ATMFFILESET.

The RECORDTX program attribute can be specified in addition to other attributes, such as COMMONTX or SEPARATETX. The other attributes are ignored if RECORDTX is specified for both the file and the program. If RECORDTX is specified for a program, but not the file, then the other options, such as COMMONTX or SEPARATETX, will be in effect.

For further details on the use of RECORDTX, see paragraph Record-Level Transactions on page 4-12.

SEPARATETX

instructs NonStop AutoTMF software to process all access to audited files in the program with separate automatic transactions.

If specified as a program attribute, SEPARATETX overrides the file attribute COMMONTX.

STOPONTMFERR

instructs AutoTMF to STOP a program that encounters a TMF environmental error. By default, AutoTMF aborts a program that encounters a TMF error. Setting STOPONTMFERR causes the program to stop without creating an Inspect saveabend file. A message describing the failure and the returned file system error code is sent to the EMS log.

See Handling TMF Environmental Errors on page 4-10 for a list of the applicable TMF errors.

By default, STOPONTMFERR is not set.

SUPPRESSINHERITEDTX

suppresses TMF transactions inherited by programs on $RECEIVE.

A non TMF-aware server may inherit a transaction from a TMF-aware requester when reading $RECEIVE. If NonStop AutoTMF software detects an inherited transaction, NonStop AutoTMF assumes that the process should use that transaction to access audited files. Suppressing inherited transactions insures that NonStop AutoTMF software generates automatic transactions for all audited file accesses, emulating the behavior of a non TMF-aware process.

Inherited transactions can also be suppressed by specifying the class map define =_ESCORT_SUPPRESS_INHRTX with a dummy file.

SUPPRESSUSERTX

suppresses the use of TMF in programs. When specified, SUPPRESSUSERTX has the following effects:

- All BEGIN and ENDTRANSACTION operations requested by the application are ignored.
- All transactions inherited from requesters through messages on $RECEIVE are ignored.
- FILEINFO requests for audited files report that files are not audited.

This facility is for testing only. Programs that make use of TMF may depend on the correct operation of the calls that are eliminated by setting this option. Enabling this option may cause program failures and data corruption.

TMF operations may be suppressed for a program by specifying the class map define =_ESCORT_SUPPRESS_AUDIT.

TXTIMEOUT { *value* | NEVER | OFF }

specifies how long automatic transactions may live, regardless of the TMF AutoAbort setting. This feature requires TMF version 3.6 or later.

- *value* is *n* [SEC[ONDS] | MIN[UTES]]
  value can be between 5 and 510 minutes and overrides the TMF AutoAbort timer.
- NEVER specifies that the TMF AutoAbort timer is not in effect.
- OFF specifies that the TMF AutoAbort is in effect.

If TXTIMEOUT is configured on a system that is not running TMF 3.6 or later, a warning message is issued, the configuration is updated and the transaction timeout takes effect when TMF 3.6 is installed.

Note. The AutoTMF AutoCommit timer always applies, regardless of the configured transaction timeout. If the desired transaction time out value exceeds the default AutoCommit timer value, the user should adjust the AutoCommit timer accordingly. See AutoCommit attribute description above on page 6-7 for details.

WAITEDIO

converts NOWAIT I/O operations to WAITED I/O operations.

If WAITEDIO is specified for a NOWAIT file open, AutoTMF automatically performs an AWAITIO operation immediately following the various forms of READ and WRITE. After the AWAITIO completes, AutoTMF commits any outstanding transactions using the usual algorithms as directed by the configuration.

AutoTMF saves the values returned by AWAITIO, and returns control to program as if the READ or WRITE operation were still in progress. For a READ, the data buffer contains the input data.

For this option to take effect, specify it both as an attribute of a program and as an attribute of the file using ADD ATMFFILESET or ALTER ATMFFILESET.

For further details, see Changing Nowait IO to Waited IO on page 4-13

ALTER ATMFFILESET

Alters the configuration of automatic transactions that were defined for a file set using the ADD ATMFFILESET command.

```
ALTER ATMFF[ILESET] file-set [ , attribute ] ... ;

attribute is

{ AUTOCOMMIT n [ SEC[ONDS] | MIN[UTES] ]| NO AUTOCOMMIT       }
{                                                             }
{ COMMONTX | NO COMMONTX                                      }
{                                                             }
{ CREATEAUDIT | NO CREATEAUDIT                                }
{                                                             }
{ HIDEAUDIT | NO HIDEAUDIT                                    }
{                                                             }
{ MAXTIME 1-to-n-seconds | NO MAXTIME                         }
{                                                             }
{ MAXUPDATES 1-to-n | NO MAXUPDATES                           }
{                                                             }
{ NOTX | NO NOTX                                              }
{                                                             }
{ NOWAIT[TX] | NO NOWAIT[TX] | WAITED[TX]                     }
{                                                             }
{ NOWARNLONGTX | NO NOWARNLONGTX                              }
{                                                             }
{ READNULLRECS                                                }
{                                                             }
{ READTHRULOCK | NO READTHRULOCK                              }
{                                                             }
{ RECORDTX | NO RECORDTX                                      }
{                                                             }
{ SEPARATETX | NO SEPARATETX                                  }
{                                                             }
{ SKIPNULLRECS                                                }
{                                                             }
{ TXTIMEOUT { value | NEVER | OFF }                           }
{                                                             }
{ WAITEDIO [ WAITTIME 1-to-4096 ] | NO WAITEDIO               }
```

file-set

TACL-style file name pattern specifying a collection of files.

attribute

AUTOCOMMIT *n* [SEC[ONDS] | MIN[UTES]] | NO AUTOCOMMIT

(*with SEPARATETX only*)

instructs NonStop AutoTMF software to commit all automatic transactions if the separate transaction for this file exceeds the specified time limit.

n is a number between 300 and 30000 but stored internally in minutes, so if specified in seconds some rounding of the value may occur. The default unit is minutes.

If *n* = 0, AUTOCOMMIT is disabled for this file set.

If a program fails to unlock a record and prevents NonStop AutoTMF software from committing automatic transactions, the non-committed updates are eventually lost because the TMF AUTOABORT timer will cause the abort of the long running transaction.

To prevent such data loss, AUTOCOMMIT should be set to a value that is less than the TMF AUTOABORT timer value, about 5 minutes or so.

The default AUTOCOMMIT value is 115 minutes, based on the default TMF AUTOABORT time limit of 120 minutes.

When AUTOCOMMIT is specified, the runtime monitors automatic transactions to determine those that have been active for more than the selected time. The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If any automatic transaction exceeds the AUTOCOMMIT time, the NonStop AutoTMF software runtime takes the following actions:

1. Force the commit of all automatic transactions,
2. Issue an EMS message,
3. Terminate the process with an abend.

The program will have to be restarted but no data will be lost.

To set this value for specific programs, use command ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

Specifying this option overrides the ATMFAUTOCOMMIT global setting.

COMMONTX | NO COMMONTX

configures files for access using automatic transactions. Use this option to manage (or not manage) file access under automatic transactions (even if the process has a transaction), for files that are not configured for SEPARATETX.

CREATEAUDIT | NO CREATEAUDIT

enables or disables (NO CREATEAUDIT) the programmatic creation of audited files.

HIDEAUDIT | NO HIDEAUDIT

requests that the audit attribute on a file be concealed from the application program when the program inquires about file attributes using procedure calls such as FILEINFO, FILERECINFO, FILEGETINFO, etc.

Be specific when defining the fileset. HIDEAUDIT hides the audit attribute for all files in the fileset, even if the audited file is not intended to be accessed under a NonStop AutoTMF software automatic transaction.

Note. If the program calls FILE_GETINFOLIST_ or FILE_GETINFOLISTBYNAME_, the audit attribute is only hidden if the check for audit is the first in the item list passed to the procedure.

MAXTIME *1-to-n-seconds* | NO MAXTIME

(*with SEPARATETX only*)

specifies that a separate transaction should be committed after *n* seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

MAXUPDATES *1-to-n* | NO MAXUPDATES

(*with SEPARATETX only*)

specifies that a separate transaction should be committed after *n* updates and inserts. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

NOTX | NO NOTX

disables or enables (NO NOTX) the generation of automatic transactions for audited file access.

NOWAIT[TX] | NO NOWAIT[TX] | WAITED[TX]

(*with SEPARATETX only*)

Enables or disables (NO NOWAIT[TX] or WAITED[TX]) the use of nowait transactions. If enabled, separate transactions are committed in nowait mode and the status of a nowait transaction is checked by the runtime when the next transaction for the file is started or when the process terminates.

NOWARNLONGTX | NO NOWARNLONGTX

(*with SEPARATETX only*)

suppresses or restores (NO NOWARNLONGTX) the warning message that is normally displayed in the EMS log when an automatic transaction runs longer than 5 minutes. The warning is repeated every 5 minutes until the transaction is committed.

Users should exercise caution when suppressing the long transaction warning and do so only if the cause for the delayed commit is well understood.

READNULLRECS

(for entry-sequenced files only)

reverses the effect of the SKIPNULLRECS option. See the description of SKIPNULLRECS below for details.

To instruct the runtime to read null records for all audited entry-sequenced files, set the global parameter ATMFSKIPNULLRECS to OFF. To request this option for a specific program, use the =_ESCORT_READ_NULL_RECS define.

If this option is specified, this specification overrides the value of the global parameter ATMFSKIPNULLRECS and of the =_ESCORT_SKIP_NULL_RECS define.

The default is READNULLRECS.

READTHRULOCK | NO READTHRULOCK

instructs NonStop AutoTMF software to enable or disable read-thru-lock mode on audited files that use automatic transactions.

If this option is specified, this specification overrides the value of the global parameter ATMFREADTHRULOCK.

To set this option for a specific program file set use ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

RECORDTX | NO RECORDTX

generates (or does not generate) an automatic transaction for each locked record and commits the transaction as soon as the record is unlocked.

RECORDTX is used when a file is accessed by programs in which record locking sequences do not allow NonStop AutoTMF software to commit automatic transactions (there is at least one lock held on a record in this file at any given time).

For this option to take effect, the option must be configured both as a program attribute and as a file attribute using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

The RECORDTX file attribute can be specified in addition to other attributes, such as COMMONTX or SEPARATETX. The other attributes are ignored if RECORDTX is specified for both the file and the program. If RECORDTX is specified for a program, but not the file, then the other options, such as COMMONTX or SEPARATETX, will be in effect.

For further details on the use of RECORDTX, see paragraph Record-Level Transactions on page 4-12.

SEPARATETX | NO SEPARATETX

enables or disables (NO SEPARATETX) the use of separate transactions by the programs to access an audited file. SEPARATETX is used when an automatic transaction is to be generated even if the process has its own transaction and when the transaction must be managed separately from transactions for other files.

SKIPNULLRECS

(for entry-sequenced files only)

instructs the NonStop AutoTMF software runtime to ignore zero-length records when reading sequentially through an audited entry-sequenced file.

If a record is inserted into an audited entry-sequenced file and subsequently backed out because a transaction is aborted, a zero length record is left in the file where the record had been inserted. This cannot occur if the file is not audited and can cause problems for programs that are not expecting to encounter such records. Configuring SKIPNULLRECS shelters the application programs from these unexpected records.

Although automatic transactions are never aborted by NonStop AutoTMF software, unilateral aborts could leave such gaps in an entry-sequenced file.

To instruct the runtime to skip null records for all audited entry-sequenced files, set the global parameter ATMFSKIPNULLRECS to OFF.

The default is READNULLRECS (null records are not skipped).

TXTIMEOUT { *value* | NEVER | OFF }

(*with SEPARATETX only*)

specifies how long automatic transactions may live, regardless of the TMF AutoAbort setting. This feature requires TMF version 3.6 or later.

- *value* is *n* [SEC[ONDS] | MIN[UTES]]
  value can be between 5 and 510 minutes and overrides the TMF AutoAbort timer.
- NEVER specifies that the TMF AutoAbort timer is not in effect.
- OFF specifies that the TMF AutoAbort is in effect.

If TXTIMEOUT is configured on a system that is not running TMF 3.6 or later, a warning message is issued, the configuration is updated and the transaction timeout takes effect when TMF 3.6 is installed.

Note. The AutoTMF AutoCommit timer always applies, regardless of the configured transaction timeout. If the desired transaction time out value exceeds the default AutoCommit timer value, the user should adjust the AutoCommit timer accordingly. See AutoCommit attribute description above on page 6-7 for details.

WAITEDIO [, WAITTIME *1-to-n-seconds*] | NO WAITEDIO

converts NOWAIT I/O operations to WAITED I/O operations.

If WAITEDIO is specified for a NOWAIT file open, AutoTMF automatically performs an AWAITIO operation immediately following the various forms of READ and WRITE. After the AWAITIO completes, AutoTMF commits any outstanding transactions using the usual algorithms as directed by the configuration.

AutoTMF saves the values returned by AWAITIO, and returns control to program as if the READ or WRITE operation were still in progress. For a READ, the data buffer contains the input data.

WAITTIME sets the maximum number of seconds an operation will wait for locks to be released. The value specified is from 1 to 4096 and rounded to the nearest power of 2 (1,2,4,8...). The default is 64 seconds.

For this option to take effect, specify it both as an attribute of a program and as an attribute of the file using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

For further details, see Changing Nowait IO to Waited IO on page 4-13.

Note. Removing an option specifying NO is different from setting it to the default value. The default value setting overrides the global (or local) value. Removing the option adopts the setting from the global (or local) setting.

ALTER ATMFPROGRAMS

Alters the configuration of automatic transactions that were defined for programs with the ADD ATMFPROGRAMS command.

```
ALTER ATMFP[ROGRAMS] object-fileset [ , attribute ] ...;

attribute is

{ AUTOCOMMIT n [ SEC[ONDS] | MIN[UTES] ] | NO AUTOCOMMIT    }
{                                                           }
{ COMMONTX | NO COMMONTX                                    }
{                                                           }
{ ISOLATION { WEAK | NORMAL | STRONG } | NO ISOLATION       }
{                                                           }
{ MAXTIME 1-to-n-seconds | NO MAXTIME                       }
{                                                           }
{ MAXTX 1-to-1000 | NO MAXTX                                }
{                                                           }
{ MAXUPDATES 1-to-n | NO MAXUPDATES                         }
{                                                           }
{ NOTX | NO NOTX                                            }
{                                                           }
{ NOWAIT[TX] | NO NOWAIT[TX] | WAITED[TX]                   }
{                                                           }
{ NOWARNLONGTX| NO NOWARNLONGTX                             }
{                                                           }
{ RECORDTX | NO RECORDTX                                    }
{                                                           }
{ OPTIMIZEUNLOCKS | NO OPTIMIZEUNLOCKS                      }
{                                                           }
{ SEPARATETX | NO SEPARATETX                                }
{                                                           }
{ STOPONTMFERR | NO STOPONTMFERR                            }
{                                                           }
{ SUPPRESSINHERITEDTX | NO SUPPRESSINHERITEDTX              }
{                                                           }
{ SUPPRESSUSERTX | NO SUPPRESSUSERTX                        }
{                                                           }
{ WAITEDIO | NO WAITEDIO                                    }
```

object-fileset

TACL-style file name pattern specifying a collection of object files

attribute

AUTOCOMMIT *n* [SEC[ONDS] | MIN[UTES]] | NO AUTOCOMMIT

instructs NonStop AutoTMF software to commit all outstanding automatic transactions if any exceeds the specified time limit.

n is a number between 300 and 30000 but stored internally in minutes, so if seconds is entered, some rounding of the value may occur. The default unit is minutes.

The default unit is minutes.

If *n* = 0, AUTOCOMMIT is disabled for this program set.

If a program fails to unlock a record and prevents NonStop AutoTMF software from committing automatic transactions, the non-committed updates are eventually lost because the TMF AUTOABORT timer will cause the abort of the long running transaction.

To prevent such data loss, AUTOCOMMIT should be set to a value that is less than the TMF AUTOABORT timer value; AUTOABORT minus 5 minutes is recommended.

The default AUTOCOMMIT value is 115 minutes, based on the default TMF AUTOABORT limit of 120 minutes.

When AUTOCOMMIT is specified, the runtime monitors automatic transactions to determine those that have been active for more than the selected time. The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If any automatic transaction exceeds the AUTOCOMMIT time, the NonStop AutoTMF software runtime takes the following actions:

1. force the commit of all automatic transactions,
2. issue an EMS message,
3. terminate the process with an abend.

The program will have to be restarted but no data will be lost.

To set this value for specific files, use command ADD ATMFFILESET or ALTER ATMFFILESET.

Specifying this option overrides the ATMFAUTOCOMMIT global setting.

COMMONTX | NO COMMONTX

configures files for access using automatic transactions. Use this option to manage file access under automatic transactions (even if the process has a transaction), for files that are not configured for SEPARATETX.

ISOLATION { WEAK | NORMAL | STRONG } NO ISOLATION

level of transaction isolation for automatic transactions:

- WEAK isolation commits outstanding transactions whenever a process replies to a request or waits for a new request. This is the default.

- NORMAL isolation commits outstanding transactions whenever a process sends a request to another process.
- STRONG isolation commits outstanding transactions whenever the process does one of the following:
  - issues a READ or a WRITE to a device,
  - issues a WRITE to a non-audited disk file

ISOLATION overrides the value of the global parameter ATMFISOLATION.

MAXTIME *1-to-n-seconds* | NO MAXTIME

specifies that transactions should be committed after *n* seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default is 16 seconds.

MAXTX *1-to-1000* | NO MAXTX

specifies the maximum number of concurrent transactions a program can process. MAXTX is rounded up to the next multiple of 100 when NonStop AutoTMF software opens the TFILE ($TMP).

The default is 100.

If an application program opens the TFILE with a transaction depth greater than 100, NonStop AutoTMF software also increases its current value of MAXTX to the value specified by the program, rounded up to the next multiple of 100.

An application program must open the TFILE before NonStop AutoTMF software opens the TFILE in order to have the maximum depth increased under program control. Once NonStop AutoTMF software has opened the TFILE, NonStop AutoTMF software cannot close and reopen TFILE with the desired depth. In such cases, you must configure MAXTX specifically if a value larger than 100 is desired.

MAXTX overrides the value set with the global parameter ATMFMAXTX.

Note. You can set MAXTX to a value greater than 100 only on NonStop servers that support a maximum of 1000 transactions (T9055AER and later)

MAXUPDATES *1-to-n* | NO MAXUPDATES

specifies that transactions should be committed after *n* updates and inserts. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default is 32.

NOTX | NO NOTX

disables or enables (NO NOTX) the generation of automatic transactions for audited file access.

NOWAIT[TX] | NO NOWAIT[TX] | WAITED[TX]

Enables or disables (NO NOWAIT[TX] or WAITED[TX]) the use of nowait transactions. If enabled, separate transactions are committed in nowait mode and the status of a nowait transaction is checked by the runtime when the next transaction for the file is started or when the process terminates.

NOWARNLONGTX | NO NOWARNLONGTX

suppresses or restores (NO NOWARNLONGTX) the warning message that is normally displayed in the EMS log when an automatic transaction runs longer than 5 minutes. The warning is repeated every 5 minutes until the transaction is committed.

Users should exercise caution when suppressing the long transaction warning and do so only if the cause for the delayed commit is well understood.

The default is NO NOWARNLONGTX.

OPTIMIZEUNL[OCKS] | NO OPTIMIZEUNL[OCKS]

specifies that UNLOCKFILE operations performed on audited files by the program fileset are suppressed. NonStop AutoTMF software considers all locks on the file released and attempts to commit the transaction, subject to the usual protocol for committing automatic transactions.

This optimization only applies to files that are enabled for automatic transactions.

UNLOCKFILE optimization can also be enabled or disabled for a program by specifying the class map define =_ESCORT_OPTIMIZEUNLOCKS or =_ESCORT_OPTMZUNLOCKSOFF.

To enable the optimization for all programs, set the global parameter ATMFOPTIMIZEUNL[OCKS] to ON.

RECORDTX | NO RECORDTX

generates (or does not generate) an automatic transaction for each locked record and commits the transaction as soon as the record is unlocked.

RECORDTX is used when a file is accessed by programs in which record locking sequences do not allow NonStop AutoTMF software to commit automatic transactions (there is at least one lock held on a record in this file at any given time).

For this option to take effect, this option must be configured both a program and a file attribute using ADD ATMFFILESET or ALTER ATMFFILESET.

The RECORDTX program attribute can be specified in addition to other attributes, such as COMMONTX or SEPARATETX. The other attributes are ignored if RECORDTX is specified for both the file and the program. If RECORDTX is specified for a program, but not the file, then the other options, such as COMMONTX or SEPARATETX, will be in effect.

For further details on the use of RECORDTX, see paragraph Record-Level Transactions on page 4-12.

SEPARATETX | NO SEPARATETX

instructs NonStop AutoTMF software to process (or not to process) all accesses to audited files in the program with separate automatic transactions.

If specified, the program attribute overrides any setting of the COMMONTX file attribute.

STOPONTMFERR | NO STOPONTMFERR

instructs AutoTMF to STOP a program that encounters a TMF environmental error. By default, AutoTMF aborts a program that encounters a TMF error. STOPONTMFERR causes the program to stop without creating an Inspect saveabend file. A message describing the failure and the returned file system error code is sent to the EMS log.

See Handling TMF Environmental Errors on page 4-10 for a list of the applicable TMF errors.

The default is NO STOPONTMFERR.

SUPPRESSINHERITEDTX | NO SUPPRESSINHERITEDTX

suppresses TMF transactions inherited by programs on $RECEIVE.

A non TMF-aware server may inherit a transaction from a TMF-aware requester when reading $RECEIVE. If NonStop AutoTMF software detects an inherited transaction, NonStop AutoTMF software assumes that the process should use that transaction to access audited files. Suppressing inherited transactions insures that NonStop AutoTMF software generates automatic transactions for all audited file accesses, emulating the behavior of a non TMF-aware process.

Inherited transactions can also be suppressed by specifying the class map define =_ESCORT_SUPPRESS_INHRTX with a dummy file.

The default is NO SUPPRESSINHERITEDTX.

SUPPRESSUSERTX | NO SUPPRESSUSERTX

suppresses the use of TMF in programs. When specified, SUPPRESSUSERTX has the following effects:

- All BEGIN and ENDTRANSACTION operations requested by the application are ignored.
- All transactions inherited from requesters through messages on $RECEIVE are ignored.
- FILEINFO requests for audited files reports that files are not audited.

This facility is for testing only. Programs that make use of TMF may depend on the correct operation of the calls that are eliminated by this option. Enabling this option may cause program failures and data corruption.

TMF operations for a program may also be suppressed by using the class map define =_ESCORT_SUPPRESS_AUDIT with a dummy file.

The default is NO SUPPTRESSUSERTX.

TXTIMEOUT { value | NEVER | OFF }

specifies how long automatic transactions may live, regardless of the TMF AutoAbort setting. This feature requires TMF version 3.6 or later.

- value is *n* [SEC[ONDS] | MIN[UTES]]
  value can be between 5 and 510 minutes and overrides the TMF AutoAbort timer.
- NEVER specifies that the TMF AutoAbort timer is not in effect.
- OFF specifies that the TMF AutoAbort is in effect.

If TXTIMEOUT is configured on a system that is not running TMF 3.6 or later, a warning message is issued, the configuration is updated and the transaction timeout takes effect when TMF 3.6 is installed.

Note. The AutoTMF AutoCommit timer always applies, regardless of the configured transaction timeout. If the desired transaction time out value exceeds the default AutoCommit timer value, the user should adjust the AutoCommit timer accordingly. See AutoCommit attribute description above on page 6-7 for details.

WAITEDIO | NO WAITEDIO

converts NOWAIT I/O operations to WAITED I/O operations.

If WAITEDIO is specified for a NOWAIT file open, AutoTMF automatically performs an AWAITIO operation immediately following the various forms of READ and WRITE. After the AWAITIO completes, AutoTMF commits any outstanding transactions using the usual algorithms as directed by the configuration.

AutoTMF saves the values returned by AWAITIO, and returns control to program as if the READ or WRITE operation were still in progress. For a READ, the data buffer contains the input data.

For this option to take effect, specify it both as a program attribute and as a file attribute using ADD ATMFFILESET or ALTER ATMFFILESET.

For further details, see Changing Nowait IO to Waited IO on page 4-13.

Note. Removing an option specifying NO is different from setting it to the default value. The default value setting overrides the global (or local) value. Removing the option adopts the setting from the global (or local) setting.

ALTER FILE

Utility command similar to the FUP ALTER command. Changes attributes of Enscribe files. Unlike the FUP ALTER command, the NonStop AutoTMF software ALTER FILE command can alter a collection of files specified as a file set.

```
ALTER FILE file-set {, specification };

specification is

{ AUDIT | NO AUDIT                                                  }
{                                                                   }
{ AUDITCOMPRESS | NO AUDITCOMPRESS                                  }
{                                                                   }
{ BUFFERED | NO BUFFERED                                            }
{                                                                   }
{ CLEARONPURGE | NO CLEARONPURGE                                    }
{                                                                   }
{ CODE file-code                                                    }
{                                                                   }
{ LOCKLENGTH key-length                                             }
{                                                                   }
{ MAXEXTENTS size                                                   }
{                                                                   }
{ NOPURGEUNTIL [ ok-time-to-purge ]                                 }
{                                                                   }
{ OWNER group-num,user-num                                          }
{                                                                   }
{ SECURE "rwep"                                                     }
{                                                                   }
{ SERIALWRITES | NO SERIALWRITES                                    }
{                                                                   }
{ VERIFIEDWRITES | NO VERIFIEDWRITES                                }
{                                                                   }
{ RESETBROKEN                                                       }
```

file-set

TACL-style file name pattern specifying a collection of files.

specification

AUDIT | NO AUDIT

specifies whether TMF auditing is on. If NO is specified, auditing is off.

The audit mode is propagated to alternate key files.

AUDITCOMPRESS | NO AUDITCOMPRESS

specifies whether or not compression of audit records is occurring for this file.

BUFFERED | NO BUFFERED

specifies whether buffered writes are performed. If NO BUFFERED is specified, writes are not buffered.

The default is buffered mode for audited files and not buffered for non audited files.

CLEARONPURGE | NO CLEARONPURGE

erases disk free space when files are purged.

CODE *file-code*

numeric file code of the file. *file-code* is an integer between 0 and 65535. Codes 100 to 999 are reserved for use by HP.

LOCKLENGTH *key-length*

the byte count of the record key for generic locks. *key-length* is between 0 and the key length of the file.

MAXEXTENTS *size*

the maximum disk allocation extents. *size* is an integer between 16 and 978, where the maximum value depends on the free space in the file label.

NOPURGEUNTIL [*ok-time-to-purge*]

date after which a PURGE of the file is allowed.

ok-time-to-purge is [*date*] *time* or [*time*] *date,*
where: *time* is *hh*:*mm*[:*ss*]
date is: *dd mmm yyyy* or *mmm dd yyyy*
(ex: 10:30:00 Jun 15 2008)

If *ok-time-to-purge* is omitted, the current time is used, thereby making the file purgeable immediately.

OWNER *group-num*, *user-num*

the user ID of the owner of the files. For example: 100,004.

RESETBROKEN

resets BROKEN flag in the file label for non audited files.

SECURE "rwep"

Guardian security string.

SERIALWRITES | NO SERIALWRITES

specifies whether serial writes to the mirrored disk are performed. If NO SERIALWRITES is specified, parallel writes are performed. The default is NO SERIALWRITES.

VERIFIEDWRITES | NO VERIFIEDWRITES

sets the mode of file writes: verified or not verified. The default is NO VERIFIEDWRITES.

ALTER GLOBAL

Sets or changes global configuration values for NonStop AutoTMF software. Global parameters are the default for all environments.

```
ALTER GLOBAL parameter-and-value;

parameter-and-value is

{ ATMF { ON | OFF }                                                    }
{                                                                      }
{ ATMFABENDNOAUDIT { ON | OFF }                                        }
{                                                                      }
{ ATMFAUDITRENAME { ON | OFF }                                         }
{                                                                      }
{ ATMFAUTOCOMMIT n [ SEC[ONDS] | MIN[UTES]  ]                          }
{                                                                      }
{ ATMFCOMMONTX  { ON | OFF }                                           }
{                                                                      }
{ ATMFISOLATION { WEAK | NORMAL | STRONG }                             }
{                                                                      }
{ ATMFMAXTIME 1-to-n-seconds                                           }
{                                                                      }
{ ATMFMAXTX 1-to-1000                                                  }
{                                                                      }
{ ATMFMAXUPDATE 1-to-n                                                 }
{                                                                      }
{ ATMFNOWAIT { ON | OFF }                                              }
{                                                                      }
{ ATMFNOWARNLONGTX { ON | OFF }                                        }
{                                                                      }
{ ATMFOPTIMIZEUNL[OCKS] | ATMFOPTMZUNL[OCKS] { ON | OFF }              }
{                                                                      }
{ ATMFREADTHRULOCK { ON | OFF }                                        }
{                                                                      }
{ ATMFSEPARATETX { ON | OFF }                                          }
{                                                                      }
{ ATMFSKIPNULLRECS { ON | OFF }                                        }
{                                                                      }
{ ATMFSTOPONTMF[ERR] { ON | OFF }                                      }
{                                                                      }
{ ATMFTXHOLDOFF 0-to-n-seconds                                         }
{                                                                      }
{ ATMFTXTIMEOUT { value | NEVER | OFF }                                }
{                                                                      }
{ DYNAMICTRACE { ON | OFF }                                            }
{                                                                      }
{ EMSCOLLECTOR { collector-process | NONE }                            }
{                                                                      }
{ MAX[MONITOR]OPENS 512-to-4096                                        }
{                                                                      }

                                                              ... more
```

```
{ SECURETRACE { ON | OFF }                                  }
{                                                           }
{ [SWAP]KMSF { ON | OFF }                                   }
{                                                           }
{ SWAPVOL[0] local-disk-volume                              }
{ SWAPVOL1 local-disk-volume                                }
{ SWAPVOL2 local-disk-volume                                }
{ SWAPVOL3 local-disk-volume                                }
```

parameter-and-value

ATMF { ON | OFF }

enables or disables automatic transaction processing. The default is ON if the system has been licensed to use NonStop AutoTMF software.

ATMFABENDNOAUDIT { ON | OFF }

specifies that NonStop AutoTMF software will abend a program that attempts to OPEN a database file that has been configured explicitly with the COMMONTX, SEPARATETX or NOTX attribute but is NOT audited at the time of the OPEN.

The default is OFF.

ATMFAUDITRENAME { ON | OFF }

enables or disables renaming of audited files.

The default is OFF.

ATMFAUTOCOMMIT *n* [SEC[ONDS] | MIN[UTES]]

instructs NonStop AutoTMF software to commit all outstanding automatic transactions and abend the process if any exceeds the specified time limit.

n is a number between 5 and 500 minutes or the equivalent number of seconds. The value is stored internally in minutes; so if seconds is entered, some rounding of the value may occur. The default unit is minutes.

If *n* = 0, AUTOCOMMIT is disabled.for all files and programs.

If a program fails to unlock a record and prevents NonStop AutoTMF software from committing automatic transactions, the non-committed updates are eventually lost because the TMF AUTOABORT timer will cause the abort of the long running transaction.

To prevent such data loss, AUTOCOMMIT should be set to a value that is less than the TMF AUTOABORT timer value: AUTOABORT minus 5 minutes is recommended because there may be some latency in detecting the long transactions.

The default AUTOCOMMIT value is 115 minutes, based on the default TMF AUTOABORT time limit of 120 minutes.

The runtime monitors automatic transactions to determine those that have been active for more than the selected time. The process must be active, either receiving messages on $RECEIVE or performing database positioning operations. Processes in a "wait" state are not monitored for long transactions.

If any automatic transaction has exceeded the AUTOCOMMIT time, the NonStop AutoTMF software runtime will take the following actions:

1. force the commit of all automatic transactions,
2. issue an EMS message,
3. terminate the process with an abend.

The program will have to be restarted but no data will be lost.

To set this value for specific files, use command ADD ATMFFILESET or ALTER ATMFFILESET and for specific programs, use command ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

ATMFCOMMONTX { ON | OFF }

enables or disables the generation of automatic transactions under a common transaction whenever the transactions are needed for audited file access.

The default is ON.

If set to OFF, you must configure the access to audited files with automatic transactions using the ADD ATMFFILESET or ADD ATMFPROGRAMS command.

The default is ON.

ATMFISOLATION { WEAK | NORMAL | STRONG }

level of transaction isolation for automatic transactions:

- WEAK isolation commits outstanding transactions whenever a process replies to a request or waits for a new request. This is the default.
- NORMAL isolation commits outstanding transactions whenever a process sends a request to another process.
- STRONG isolation commits outstanding transactions whenever the process does one of the following:
  - issues a READ or a WRITE to a device
  - issues a WRITE to a non-audited disk file

ISOLATION can also be set for selected programs using the ADD ATMFPROGRAMS command.

ATMFMAXTIME *1-to-n-seconds*

specifies that transactions should be committed after *n* seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default is 16 seconds.

MAXTIME can also be set for separate transactions and for selected programs. See ADD ATMFFILESET and ADD ATMFPROGRAMS for details.

ATMFMAXTX *1-to-1000*

specifies the maximum number of concurrent transactions each prepared program can process.

ATMFMAXTX is rounded up to the next multiple of 100 when NonStop AutoTMF software opens the TFILE ($TMP).

The default is 100.

If an application program opens the TFILE with a transaction depth greater than 100, NonStop AutoTMF software also increases its current value of MAXTX to the value specified by the program, rounded up to the next multiple of 100.

An application program must open the TFILE before NonStop AutoTMF software opens the TFILE to have the maximum depth increased under program control. Once NonStop AutoTMF software has opened the TFILE, it cannot close and reopen the TFILE with the desired depth.

In such cases, you must configure MAXTX specifically if a value larger than 100 is desired.

To override ATMFMAXTX for a specific program, set the MAXTX option using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

Note. You can set ATMFMAXTX to a value greater than 100 only on NonStop servers that support a maximum of 1000 transactions (T9055AER and later)

ATMFMAXUPDATE *1-to-4096*

specifies that transactions should be committed after *n* updates and inserts. The value of *n* should be a power of 2 (1,2,4,8,16,32,...) in the range of 1 to 4096. If the value specified is not a power of 2, the value will be rounded to a power of 2.

The default value is 32 updates and inserts.

MAXUPDATES can also be set for separate transactions and for selected programs. See ADD ATMFFILESET and ADD ATMFPROGRAMS for details

ATMFNOWAIT { ON | OFF }

returns control to the application without waiting for the transaction to be fully committed. Automatic transactions are committed in nowait mode. The status of a nowait transaction is checked by the runtime when the next transaction for the file is started or when the process terminates.

The default is OFF.

NOWAIT can also be set for separate transactions and for selected programs. See ADD ATMFFILESET and ADD ATMFPROGRAMS for details.

ATMFNOWARNLONGTX { ON | OFF }

suppresses the warning message that is normally displayed in the EMS log when an automatic transaction runs longer than 5 minutes. The warning is repeated every 5 minutes until the transaction is committed.

Users should exercise caution when suppressing the long transaction warning and do so only if the cause for the delayed commit is well understood.

The default is OFF.

ATMFOPTIMIZEUNL[OCKS] | ATMFOPTMZUNL[OCKS] { ON | OFF }

specifies that UNLOCKFILE operations performed on audited files by an application process are eliminated. NonStop AutoTMF software considers all locks on the file released and attempts to commit the transaction, subject to the usual protocol for committing automatic transactions.

This optimization only applies to files that are enabled for automatic transactions.

The default is OFF and can be overridden for specific programs by using ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS.

ATMFREADTHRULOCKS { ON | OFF }

instructs NonStop AutoTMF software to enable or disable read-thru-lock mode on audited files that use automatic transactions.

The default is ON.

The value can be overridden for a program file set using the ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS and for a file set using the ADD ATMFFILESET or ALTER ATMFFILESET.

ATMFSEPARATETX

instructs NonStop AutoTMF software to (or not to) generate a separate parallel transaction for all files under its control.

ATMFSEPARATETX should be used with great caution as it might have a measurable negative impact on performance.

The default is OFF.

ATMFSKIPNULLRECS

instructs the NonStop AutoTMF software runtime to ignore zero-length records when reading sequentially through an audited entry-sequenced file.

If a record is inserted into an audited entry-sequenced file and subsequently backed out because a transaction is aborted, a zero-length record is left in the file where the record had been inserted. This cannot occur if the file is not audited and can cause problems for programs that are not expecting to encounter such records. Setting ATMFSKIPNULLRECS to ON shelters all programs from reading unexpected null records in all entry-sequenced files.

Although automatic transactions are never aborted by NonStop AutoTMF software, unilateral aborts could leave such gaps in an entry-sequenced file.

The default is OFF.

ATMFSTOPONTMF[ERR]

instructs AutoTMF to STOP a program that encounters a TMF environmental error. By default, AutoTMF aborts a program that encounters a TMF error. Setting this option causes the program to stop without creating an Inspect saveabend file. A message describing the failure and the returned file system error code is sent to the EMS log.

See Handling TMF Environmental Errors on page 4-10 for a list of the applicable TMF errors.

The default is OFF.

ATMFTXHOLDOFF *0-to-n-seconds*

Keeps automatic transactions active between requests received by a server if the time between these requests is smaller than *n* seconds. The value of *n* should be between 0 and 120.

The default is 0 seconds and automatic transactions are committed when a server process replies to a request.

This feature is especially useful for certain types of batch processing that are implemented as a master process that sends requests to server processes. In these cases, if the batch process is restarted after a failure, some completed server requests may be aborted.

When the master process stops sending work to the server, there may be an outstanding active transaction, plus record locks and uncommitted updates. A commit timer will commit the active transactions after a specified idle period. The master process may send a terminating request to the servers but should not stop the server processes.

Automatic transactions are committed in any of the following instances:

- the process terminates (but not if the process is stopped).

- the server is waiting for a request and the time since the last request exceeds the specified value.
- the transaction reaches the limits set by the MAXTIME and MAXUPDATES parameters.

Each transaction spans many server requests and updated records are kept locked between server requests. If the batch process sends work to many server processes, the servers processes must access different database records or a deadlock may occur. The commit timer will not cause the current transaction to be committed if the process is deadlocked.

ATMFTXTIMEOUT { value | NEVER | OFF }

specifies how long automatic transactions may live, regardless of the TMF AutoAbort setting. This feature requires TMF version 3.6 or later.

- *value* is *n* [SEC[ONDS] | MIN[UTES]]
  value can be between 5 and 510 minutes and overrides the TMF AutoAbort timer.
- NEVER specifies that the TMF AutoAbort timer is not in effect.
- OFF specifies that the TMF AutoAbort is in effect.

If TXTIMEOUT is configured on a system that is not running TMF 3.6 or later, a warning message is issued, the configuration is updated and the transaction timeout takes effect when TMF 3.6 is installed.

Note. The AutoTMF AutoCommit timer always applies, regardless of the configured transaction timeout. If the desired transaction time out value exceeds the default AutoCommit timer value, the user should adjust the AutoCommit timer accordingly. See AutoCommit attribute description above on page 6-7 for details.

DYNAMICTRACE { ON | OFF }

enables or disables dynamic tracing of prepared programs. If enabled, processes periodically check a signaling file to determine if tracing should be turned on or off.

The default is ON.

EMSCOLLECTOR { *collector-process* | NONE }

alternate collector process for runtime EMS event messages generated by the NonStop AutoTMF software runtime library. EMS events are internal and generally related to configuration problems.

collector-process specifies an EMS collector, having a device type 1 and subtype 0; otherwise, the command is ignored.

NONE means EMS logging will not occur.

MAX[MONITOR]OPENS *512-to-4096*

specifies the default maximum number of simultaneous open requests supported by the monitor process.

The default is 512.

SECURETRACE ON | OFF

controls tracing of processes by users other than the owner of the process.

The default is ON.

[SWAP]KMSF { ON | OFF }

enables or disables KMSF swapfile allocation.

If an error occurs on the swapfile allocation, the runtime reverts to the normal swapfile allocation, either using the SWAPVOLn global specification or the default swapfile.

The default is OFF.

SWAPVOL[0] *local-disk-volume*
SWAPVOL1 *local-disk-volume*
SWAPVOL2 *local-disk-volume*
SWAPVOL3 *local-disk-volume*

names the extended-memory, swap file volumes for the NonStop AutoTMF software runtime library private segment.

If *local-disk-volume* is omitted, the SWAPVOL[n] setting is deleted.

When a program is launched an extended memory segment and its disk swap files are allocated. The specified volumes are used randomly: all are tried until one succeeds. If the volumes all fail, the process default swap volume is also tried.

ALTER MAPDB

Alters MapDB attributes; most of these attributes pertain to the monitor process, not the mapping database. Changes take effect when the monitor process is restarted.

These attributes can also be changed using the ALTER MONITOR command.

```
ALTER MAPDB [subvol] { , option } ;

option is

{ BACKUP cpu-number                                                }
{                                                                  }
{ HOMETERM [ file-name | NONE ]                                    }
{                                                                  }
{ MONITOR process-name                                             }
{                                                                  }
{ PRIMARY cpu-number                                               }
{                                                                  }
{ PRIORITY process-priority                                        }
```

subvol

name of MapDB subvolume. The default is the current CI session MapDB.

option

BACKUP *cpu-number*

cpu (in the range of 0 to 15) of the backup monitor processes. Used when the monitor process is started.

HOMETERM *file-name*

home terminal for the monitor process.

If omitted, the home terminal is set to the default EMS collector. The default EMS collector is $0, or the EMS collector configured by setting the global parameter EMSCOLLECTOR.

Note that internally, the default hometerm of the monitor is $ZHOME, as displayed by the TACL command STATUS $<*process-name*>, because an EMS collector cannot be specified as the hometerm of a process when the process is launched. Functionally however, the home terminal for the monitor process is the default EMS collector, as shown by the AutoTMF STATUS MONITOR command.

MONITOR *process-name*

NonStop AutoTMF software monitor process to associate with the MapDB subvolume. The name must be 5 characters long including the $. The default is $ZESC.

PRIMARY cpu-number

cpu (in the range of 0 to 15) of the primary monitor processes. Used when the monitor process is started.

PRIORITY *process-priority*

priority (in the range of 1 to 199) for the monitor process. Used when the monitor process is started.

ALTER MONITOR

Alters attributes of the monitor process. Changes take effect when the monitor process is restarted.

These attributes can also be changed using the ALTER MAPDB command.

```
ALTER MONITOR [process-name] { , option };

option is

{ BACKUP cpu-number                                                    }
{                                                                      }
{ CACHEKB size                                                         }
{                                                                      }
{ HOMETERM [file-name | NONE ]                                         }
{                                                                      }
{ PRIMARY cpu-number                                                   }
{                                                                      }
{ PRIORITY process-priority                                            }
```

process-name

process name of the NonStop AutoTMF software monitor process. The name must have 5 characters, including the $. The default is the current CI session monitor.

option

BACKUP *cpu-number*

cpu (in the range of 0 to 15) of the backup monitor processes. Used when the monitor process is started.

CACHEKB *size*

specifies in kilobytes the amount of memory to be allocated for caching NonStop AutoTMF software file and program attributes. The size ranges from 100 kilobytes to 127 megabytes, the maximum extended segment size. Approximately 32K of this cache is reserved for internal uses.

HOMETERM *file-name*

home terminal for the monitor process.

If omitted, the home terminal is set to the default EMS collector. The default EMS collector is $0, or the EMS collector configured by setting the global parameter EMSCOLLECTOR.

Note that internally, the default hometerm of the monitor is $ZHOME, as displayed by the TACL command STATUS $<*process-name*>, because an EMS collector cannot be specified as the hometerm of a process when the

process is launched. Functionally however, the home terminal for the monitor process is the default EMS collector, as shown by the AutoTMF STATUS MONITOR command.

PRIMARY *cpu-number*

cpu (in the range of 0 to 15) of the primary monitor processes. Used when the monitor process is started.

PRIORITY *process-priority*

priority (in the range of 1 to 199) for the monitor process. Used when the monitor process is started.

ALTER LOCAL

Sets or changes a local parameter value. Local parameters are the same as global parameters and have the same purpose, but they are in effect for the current Monitor (MapDB) only. Local parameter values are stored in the MapDB table REGISTRY. Setting a local value overrides the global (default) value.

To restore the original default value of the parameter, use RESET LOCAL.

For a description of the local parameter values, see command description for ALTER GLOBAL on page 6-34.

CALC

A four-function calculator. Operators include addition, subtraction, multiplication, division and parentheses. Multiplication and division have precedence over addition and subtraction. Numeric operands are assumed to be decimal unless preceded by % (for octal), %h (for hexadecimal), or %b (for binary). An expression comprised of simple integers is processed using integer arithmetic. If any operand contains a decimal point, the expression will be evaluated to four decimal places of accuracy.

```
CALC expression;
```

expression

is one of:

- a *constant*
- *expression operator expression*
- *expression*

constant

specifies a decimal constant with or without a decimal point, or a binary, octal, or hexadecimal integer prefixed by %b, % or %h respectively.

operator

the arithmetic operator +, -, * or /

COMMENT (or “--”)

Causes the command interpreter to ignore the remainder of the current line. COMMENT may not appear within the lines of a multiline command.

In addition to the explicit COMMENT command, a pair of dashes (– –) causes the CI to ignore all remaining text on the current line. A dash-dash comment may appear within a multiline command.I

```
COMMENT any-text
or
-- any-text
```

COPY

Similar to the FUP COPY command with enhancements. Copies records from an input file to an output file.

```
COPY in-file, out-file [, copy-options ];

copy-options is:
{ control-options                                                    }
{ in-options                                                         }
{ out-options                                                        }
{ display-options                                                    }

control-options is:
{ COUNT num-records                                                  }
{                                                                    }
{ FIRST { ordinal-record-num |                                       }
{ { KEY {record-spec|key-value [, key-value]}|                       }
{ { key-spec ALTKEY key-value [, key-value ] }                       }
{                                                                    }
{ FROMLAST                                                           }
{                                                                    }
{ UPSHIFT                                                            }

in-options is:

{ BLOCKIN n                                                          }
{                                                                    }
{ COMPACT | NO COMPACT                                               }
{                                                                    }
{ COMP[ARELEN] n                                                     }
{                                                                    }
{ EBCDICIN                                                           }
{                                                                    }
{ EXACT                                                              }
{                                                                    }
{ RECIN n                                                            }
{                                                                    }
{ REVERSE                                                            }
{                                                                    }
{ REWINDIN | NO REWINDIN                                             }
{                                                                    }
{ SKIPMATCH                                                          }
{                                                                    }
{ TRIM trim-character                                                }
{                                                                    }
{ UNLOADIN | NO UNLOADIN                                             }
{                                                                    }
{ UNSTRUCTURED                                                       }
{                                                                    }
{ VARIN
                                                          ...more
```

```
out-options is:

{ BLOCKOUT n                                                    }
{                                                               }
{ EBCDICOUT                                                     }
{                                                               }
{ FOLD                                                          }
{                                                               }
{ PAD pad-character                                             }
{                                                               }
{ RECOUT n                                                      }
{                                                               }
{ REWINDOUT | NO REWINDOUT                                      }
{                                                               }
{ UNLOADOUT | NO UNLOADOUT                                      }
{                                                               }
{ UNSTROUT                                                      }
{                                                               }
{ UPDATE                                                        }
{                                                               }
{ VAROUT                                                        }

display-options

{ [O]CTAL                                                       }
{                                                               }
{ [D]ECIMAL                                                     }
{                                                               }
{ [H]EX                                                         }
{                                                               }
{ [A]SCII                                                       }
{                                                               }
{ BYTE                                                          }
{                                                               }
{ NO HEAD                                                       }
```

in-file

file containing data to be copied; can be a process, tape, terminal, or disk file. Disk files include edit files, Enscribe structured and unstructured files. If the in-file is omitted, the CI in file is used.

out-file

file to receive data to be copied; can be a process, tape, terminal, a printer or disk file. Disk files include edit files, Enscribe structured and unstructured files. If the out-file is omitted, the CI out file is used.

control-options

COUNT *num-records*

the number of records or rows to be copied. If omitted, all records are copied.

FIRST { *ordinal-record-num* }
{ KEY { *record-spec* | *key-value* } }
{ *key-spec* ALTKEY *key-value* [, *key-value*] }

the starting record of the input file to copy. If omitted, the copy starts at the first record or row in the input file.

ordinal-record-num

the number of records or rows from the beginning of the file that are to be skipped. The first record in a file is record zero. The maximum value is 4294967295.

KEY { *record-spec* | *key-value* }

the primary key value for the starting record or row of a structured disk file.

record-spec is a fixed quantity limited to 2**63 -1.

- for unstructured files, *record-spec* is the starting relative byte address
- for relative files, *record-spec* is the starting record number
- for entry-sequenced files, *record-spec* iis the 64-bit numeric record key

key-value applies only to key-sequenced files and specifies the approximate position of the starting record; *key-value i*s specified as a string or as integer byte values in the range of 0 to 255.

The key-value is entered as follows:

"[" { string } [, { string }] "]"

{ 0:255 } [{ 0:255 }]

where the integers represent the byte values. To specify a list of strings, enclose each string in quotation marks separated by a comma and enclose the list in square brackets.

key-spec

the alternate key tag (a 2-byte string or a 16-bit integer) designating the alternate key to be used for positioning.

ALTKEY *key-value* [, *key-value*]

the alternate key of the starting record or row. The format of *key-value* is described above.

FROMLAST

position on the last record in the key range specified in the FIRST KEY option.

UPSHIFT

convert lowercase characters to uppercase.

in-options

BLOCKIN *n*

number of bytes between 1 and 32767 in an input block that is requested in a single physical read operation. When BLOCKIN is not specified, the RECIN value is used. The default is device dependent: 80 bytes for terminal, 132 bytes for process and unstructured files.

COMPACT | NO COMPACT

zero length records should (or should not) be skipped when copied to the output file. The default is COMPACT.

COMPACT applies only for copying relative files

COMP[ARELEN] *n*

use generic positioning for compare length *n* on the record key (primary or alternate) specified in the FIRST KEY option. The compare length is between 1 and 255 and must be less than or equal to the key specified.

EBCDICIN

translate input characters from EBCDIC to ASCII.

EXACT

exact positioning on the record key (primary or alternate) specified in the FIRST KEY option.

RECIN *n*

the maximum number of bytes in an input record. *n* is between 1 and 4096. When RECIN is not specified, the BLOCKIN value is used with a maximum of 4096.

REVERSE

reads the input file from the starting record in reverse order.

REWINDIN | NO REWINDIN (*magnetic tapes only*)

input tape is rewound (or not rewound) when the EOF is read from the tape. If NO REWINDIN is specified, the tape remains positioned without rewinding.The default is REWINDIN. This option also applies to labeled tapes.

SHARE

the file is to be opened in shared exclusion mode. The default is protected.

SKIPMATCH

position the input file to the record immediately following the one whose key matches the specified key. The entire key must be supplied. If the file is not key sequenced, an error 46 is returned.

This option applies to key sequenced files only.

TRIM *trim-character*

delete any trailing character matching the *trim-character*. The character is specified in ASCII using quotation marks or as an integer in the range 0 to 255.

UNLOADIN | NO UNLOADIN (*magnetic tapes only*)

input tape is unloaded (or not unloaded) after the tape has been rewound. The default is UNLOADIN. This also applies to labeled tapes.

UNSTRUCTURED

open and access the input file using the unstructured option. This option can be used for Enscribe unstructured or structured files where the file structure is ignored.

VARIN

read variable length blocked records. These records can be produced by using the VAROUT COPY command option described below. Each record is preceded by a one word indicator that contains the record length in bytes.

out-options

BLOCKOUT *n*

number of bytes from 1 to 32767 in an output record. When BLOCKOUT is not specified, the RECOUT value is used. The default is device dependent: 80 bytes for terminal, 132 bytes for a process and unstructured files. If BLOCKOUT is greater than RECOUT, the output block is filled with RECOUT-value length records until the block contains BLOCKOUT-value bytes or the last output record is encountered.

EBCDICOUT

translate output characters from ASCII to EBCDIC.

FOLD

output records longer than the output record length will be divided into as many output records as needed to copy the entire record.

PAD *pad-character*

output records shorter than the output record length will be padded with *pad-character*. Specify *pad-character* as an ASCII character in quotation marks or as an integer in the range of 0 to 255.

RECOUT *n*

maximum length between 1 and 4096 of an output record.

REWINDOUT | NO REWINDOUT (*magnetic tapes only*)

output tape is rewound (or not rewound) after the copy operation has completed. If NO REWINDOUT is specified, the tape remains positioned without rewinding. The default is REWINDOUT. This option also applies to labeled tapes.

UNLOADOUT | NO UNLOADOUT (*magnetic tapes only*)

output tape is unloaded (or not unloaded) after rewinding. The default is UNLOADOUT. This option also applies to labeled tapes.

UNSTROUT

opens and writes the output file using the unstructured access option. This option is used for Enscribe unstructured files or structured files where the file structure is ignored.

UPDATE

records are updated rather than inserted into the output file. An error is returned if the record to update is not present in the file.

This option is valid for key sequenced files only. If the output file is not key sequenced, the option is ignored.

VAROUT

write variable length blocked records. Each record is preceded by a one word indicator that contains the record length in bytes.

Variable length records are word aligned in the output block. The last record in each block is followed by a terminator (-1) if there is space in the block.

The FOLD and PAD options are not supported when varout is specified.

display-options

Display options differ slightly from FUP display options: output data transformations (RECOUT, BLOCKOUT, PAD and so on) are applied before formatting the data. Thus, formatted data is displayed as the data would be written to a disk file.

Entry-sequenced record keys are displayed in the 64-bit format.

[O]CTAL

display the output in octal and ASCII format.

[D]ECIMAL

display the output in decimal and ASCII format

[H]EX

display the output in hexadecimal and ASCII format

[A]SCII

display the output in ASCII format. This option is ignored if combined with OCTAL, HEX, DECIMAL or BYTE display options.

NO HEAD

omit the heading preceding each record when one of the *display-options* is specified.

CPUS

Displays the CPU configuration of a node.

```
CPUS [ node-name ];
```

node-name

a node in the Expand network. The default is the local system

CREATE MAPDB

Creates a set of SQL tables used to configure NonStop AutoTMF software and configures the associated NonStop AutoTMF software monitor process. See Monitor Process on page A-2 for guidelines about managing the monitor process.

Because MapDB contains SQL tables, the MapDB subvolume must be created on an audited volume.

```
CREATE MAPDB [ subvol  | ON volume ] [ , option ] ;

option is

{ CATALOG SQL-catalog                                                  }
{                                                                      }
{ SECURE "rwep"                                                        }
{                                                                      }
{ MONITOR process-name                                                 }
{                                                                      }
{ PRIMARY cpu-number                                                   }
{                                                                      }
{ BACKUP cpu-number                                                    }
{                                                                      }
{ PRIORITY process-priority                                            }
{                                                                      }
{ HOMETERM file-name                                                   }
```

subvol

subvolume where MapDB tables are to be created.

volume

volume on which MapDB tables are to be created. The MapDB subvolume is ESCMAPDB.

option

CATALOG *SQL-catalog*

SQL catalog where MapDB tables are to be registered. The catalog may also be specified either by the VOLUME command or by specifying a catalog attribute for the =_DEFAULTS define.

If omitted, the tables are registered in the catalog where SysDB tables are registered.

SECURE *"rwep"*

specifies the READ, WRITE, EXECUTE and PURGE security attributes of MapDB tables.

MONITOR *process-name*

NonStop AutoTMF software monitor process to be associated with MapDB. The name must have 5 characters, including the $. The default is $ZESC.

PRIMARY *cpu-number*

cpu (in the range of 0 to 15) of the primary monitor processes. Used when the monitor process is started.

BACKUP *cpu-number*

cpu (in the range of 0 to 15) of the backup monitor processes. Used when the monitor process is started.

PRIORITY *process-priority*

priority (in the range of 1 to 199) for the monitor process. Used when the monitor process is started.

HOMETERM *file-name*

home terminal for the monitor process.

If omitted, the home terminal is set to the default EMS collector. The default EMS collector is $0, or the EMS collector configured by setting the global parameter EMSCOLLECTOR.

Note that internally, the default hometerm of the monitor is $ZHOME, as displayed by the TACL command STATUS $<*process-name*>, because an EMS collector cannot be specified as the hometerm of a process when the process is launched. Functionally however, the home terminal for the monitor process is the default EMS collector, as shown by the AutoTMF STATUS MONITOR command.

When a MapDB is created, the monitor process is started to provide access to MapDB tables. Operational procedures should be updated to start the monitor after a system cold load.

CREATE SYSDB

Creates the system database (SysDB). SysDB is a set of SQL tables in a subvolume named ESCSYSDB. These tables are updated with registration information and the list of MapDBs and associated monitor processes.

SysDB must be created on an audited volume.

```
CREATE SYSDB [ ON volume ] [, option ];

option is

{ CATALOG SQL-catalog                                    }
{                                                        }
{ SECURE "rwep"                                          }
```

volume

volume where SysDB will be created. The default is $SYSTEM.

option

CATALOG *SQL-catalog*

SQL catalog where SysDB tables are to be registered. The catalog may also be specified either by the VOLUME command or by specifying a catalog attribute for the =_DEFAULTS define.

If not specified, an SQL catalog is created on the same volume as SysDB in a subvolume called ESCCATLG.

SECURE *"rwep"*

specifies the READ, WRITE, EXECUTE and PURGE security attributes of SysDB tables.

After you create SysDB, comply with the product licensing instructions.

Do not create more than one SysDB, because all disks are searched for SysDB subvolumes (named ESCSYSDB) and an error occurs if more than one is found.

DEADLOCK

Creates a record lock deadlock on an Enscribe file. The command is used to test deadlock detection of the LISTLOCKS command. This command does not change the file.

```
DEADLOCK file [,DEPTH 2-to-n] [,TIME 2-to-n-seconds];
```

file

specifies the name of an Enscribe file on which the deadlock situation is to be created.

DEPTH *2-to-n*

the number of lock holders (file opens or transactions) between 2 and 10 that participate in the deadlock. Each holder will be waiting for a unique record locked by another holder, creating an *n*-way deadlock. Additional records are locked that are not material to the deadlock.

The default depth is 2.

TIME 2-to-n-seconds

the duration of the deadlock in seconds where *n* is between 2 and 600. The deadlock can be stopped by entering the break key.

The default is 60 seconds.

Note. If any transactional activity has occurred in ESCORT CI before the DEADLOCLK command is issued, DEADLOCK fails with an error 13 on the OPEN of $TMP. Transactional activity includes any configuration command that requires access to SysDB or MapDB. If this error occurs, exit and restart the ESCORT CI session.

DELAY

Suspends the execution of the AutoTMF command interpreter for the specified interval for the specified interval or until the BREAK key is pressed.

```
DELAY { n-centisecs | n SEC[ONDS] | n MIN[UTES] };
```

DELETE ATMFFILESET

Removes the configuration of a file set defined with the ADD ATMFFILESET command.

```
DELETE ATMFF[ILESET] file-set;
```

file-set

TACL-style file name pattern specifying a collection of files.

DELETE ATMFPROGRAMS

Removes the configuration of a program file set defined with the ADD ATMFPROGRAMS command.

```
DELETE ATMFP[ROGRAMS] object-fileset;
```

object-fileset

TACL-style file name pattern specifying a collection of files.

DEQUEUE

Reads records from a source Enscribe queue file and writes them to a destination file. DEQUEUE syntax and functionality are is similar to the COPY command. The main difference between the commands is that instead of a READ of the source file, it uses READUPDATELOCK operations that dequeue (and therefore delete) each record as it is read from the queue file.

Full transaction support is provided. If both input and output files are audited, one transaction provides consistency of dequeues and inserts.

The DEQUEUE command terminates if the queue is empty. It times out after two seconds if a read on the queue can't find a record.

```
DEQUEUE in-file, out-file [, dequeue-options ];

dequeue-options is:

{ control-options                                                    }
{ in-options                                                         }
{ out-options                                                        }
{ display-options                                                    }
```

in-file

an Enscribe queue file.

out-file

an Enscribe queue file, entry-sequenced file, the terminal or $NULL (or equivalent).

dequeue-options

See command options for COPY on page 6-46.

DROP MAPDB

Deletes an entire MapDB; use this command with caution.

```
DROP MAPDB subvolume
```

subvolume

an existing MapDB subvolume.

ENV

Displays the current setting of all environmental variables.

```
ENV;
```

EXIT

Terminates the CI session. A CTRL–Y has the same effect.

```
EXIT
```

The EXIT command does not require a semi-colon and cannot be followed by any text.

FACTOR

Calculates the prime factors of the integer part of the expression. Accepts any numeric expression.

```
FACTOR constant;
```

constant

an ordinary decimal constant with or without a decimal point, or a binary, octal, or hexadecimal integer prefixed by %b, %, or %h respectively.

FC and !

Runs previous commands found in the command history. FC permits a command to be edited before execution. “!” runs a command without editing. The commands are not multiline commands and are not terminated with a semicolon.

```
FC [ integer | -integer | text ]

! [ integer | -integer | text ]
```

The desired command can be specified in one of four ways:

- The default is the immediately previous command;
- *integer* (positive) specifies the ordinal number of a command in the history buffer (See HISTORY).
- *-integer* (negative) specifies a relative command in the history buffer, with the most recent command having the value –1.
- *text* selects the most recent command that starts with the specified text.

FILEINFO

Displays information about Enscribe files.

```
FI[LEINFO] file-set [, DETAIL ];
or
FID file-set;
```

file-set

is a TACL-style file name pattern specifying a collection of files. If *file-set* is not specified, the current subvolume is assumed.

DETAIL

displays detailed information. If not specified, the CI displays one line of information per file in the file-set.

FID *file-set*

abbreviates FILEINFO DETAIL.

FILES

Displays the 8-character filename of the files in the file-set.

```
FILES file-set;
```

file-set

is a TACL-style file name pattern specifying a collection of files. If *file-set* is not specified, the current subvolume is assumed.

HELP

Lists the help options for NonStop AutoTMF software commands, MONITOR commands and the NonStop AutoTMF software defines.

```
HELP
{ ALL                                                          }
{                                                              }
{ AUTOTMF-COMMAND                                              }
{                                                              }
{ autotmf-command [ DETAIL | EXAMPLE ]                         }
{                                                              }
{ AUTOTMF-DEFINES                                              }
{                                                              }
{ autotmf-define [ USAGE ]                                     }
{                                                              }
{ GLOBALS                                                      }
{                                                              }
{ MONITOR [ monitor-command ]                                  }
{                                                              }
{ TRACING [DETAIL | EXAMPLE]                                   }
```

ALL

displays the list of all commands.

AUTOTMF-COMMANDS

requests the list of all NonStop AutoTMF software commands available.

autotmf-command [DETAIL | EXAMPLES]

is a NonStop AutoTMF software command. Help displays the syntax and description of the command.

Multi-word commands are entered with hyphens.

For example, to obtain help on ADD AUTOTMFFILESET, type:

HELP add-autotmffileset;

If DETAIL is specified, a description of the command parameters is displayed in addition to the syntax.

If EXAMPLES is specified, Help displays examples of the command.

If neither DETAIL nor EXAMPLES is specified, Help displays both as possible subtopics.

MONITOR [*monitor-command*]

displays the syntax and description of commands that can be sent to the NonStop AutoTMF software monitor process via the CI. If no monitor command is specified, HELP displays a list of the commands as subtopics.

monitor-commands is one of: BACKUPCPU, LOG, SECURITY, STATUS or SWITCH or TRACE.

AUTOTMF-DEFINES

requests the display of the list of DEFINEs for which HELP is available.

define-name [USAGE]

is the name of the DEFINE. HELP displays the syntax and a description of the DEFINEs.

If USAGE is specified, Help displays information about the usage of the DEFINE, otherwise Help displays USAGE as a possible subtopic.

GLOBALS

requests a display of the list of all the possible global parameters that can be set for NonStop AutoTMF software using the ALTER GLOBAL command.

TRACING

requests information on tracing options and the commands to use to initiate tracing.

Note. If the Monitor process is not running, the output of the HELP command may be incomplete for certain commands or topics.

HISTORY

Lists the saved commands in the history buffer. These commands can be run using the FC or bang (!) commands.

```
HISTORY count;
```

count

the number of commands to display. The default is 10. If fewer commands are in the history buffer, all commands will be displayed.

INFO ATMFFILESET

Displays a configuration for a file set defined with the ADD ATMFFILESET command.

```
INFO ATMFF[ILESET] [ file-set ] [, OBEYFORM ];
```

file-set

name of the file set that was configured with ADD ATMFFILESET.

If file-set is not specified, the entire ATMFFILESET configuration is displayed.

To view the NonStop AutoTMF software configuration for specific files, use the INFO LIBRARY command with the DETAIL option.

OBEYFORM

displays the fileset configuration as ADD ATMFFILESET commands.

INFO ATMFPROGRAMS

Displays information about special NonStop AutoTMF software attributes that are configured for a program fileset using the ADD ATMFPROGRAMS command.

```
INFO ATMFP[ROGRAMS] [ object-fileset ] [, OBEYFORM ];
```

object-fileset

name of the fileset configured with ADD ATMFPROGRAMS.

If *object-fileset* is not specified, the entire ATMFPROGRAMS configuration is displayed.

To view the NonStop AutoTMF software configuration for specific programs, use the INFO LIBRARY command with the DETAIL option or the INFO PROGRAM and PROGINFO (PI) commands with the DETAIL option.

OBEYFORM

displays of the configuration as ADD ATMFPROGRAMS commands.

INFO GLOBALS

Displays the list of configured NonStop AutoTMF software global parameters.

```
INFO GLOBAL[S] [,OBEYFORM];
```

OBEYFORM

displays a list of the ALTER GLOBAL commands that were entered to modify the default global values.

See command ALTER GLOBAL above for a list of global parameters and values.

INFO LIBRARY

Displays a list of processes that use the specified file as a user library.

```
INFO LIBRARY filename [, STOP ];
```

filename

the name of a user library.

STOP

Stops all running processes that use *filename* as a user library.

The processes is first listed and you are asked to confirm “Do you wish to stop *n* processes (Y[ES] or N[O])?” The command then displays the current process list, stopping each one. The two lists may not be the same if processes start or stop while awaiting confirmation

Use this command with caution. The STOP option is useful for the system administrator when installing a new version of the NonStop AutoTMF software runtime library to stop the application processes that currently use the library.

INFO LOCALS

Displays the list of configured NonStop AutoTMF software local parameters. Local parameters are the default settings for a specific MapDB and monitor. They are the same as global parameters.

```
INFO LOCAL[S] [, OBEYFORM];
```

OBEYFORM

displays a list of the ALTER GLOBAL commands that were entered to modify the default global values.

See command ALTER GLOBAL on page 6-34 above for a list of global parameters and values.

INFO MAPDB

Displays information about the MapDBs in SysDB.

```
INFO MAPDB { * | subvolume } [, DETAIL ];
```

*

displays information for all configured MapDBs.

subvolume

an existing MapDB subvolume.

INFO PREPARE

Lists the external procedure names that are retargeted by the PREPARE command and intercepted by the NonStop AutoTMF software runtime library.

```
INFO PREPARE;
```

INFO PROGRAM

Displays information about object files. If NonStop AutoTMF software program attributes are configured for an object file in the file set specified, these attributes are also displayed.

```
INFO PROGRAM [file-set ] [, AFTER filename | START filename ]
             [, prep-info-option | object-info-option ];

prep-info-option is

{ DETAIL                                                      }
{                                                             }
{ PREPARED                                                    }

{ UNPREPARED                                                  }

object-info-option is

{ AXCELCHECK                                                  }
{                                                             }
{ COBOL                                                       }
{                                                             }
{ CONFLICTS                                                   }
{                                                             }
{ PROCS | PROCEDURES [, AXL ] | [, OCA ]                      }
{                                                             }
{ PROG[RAMS]                                                  }
{                                                             }
{ REBASE                                                      }
{                                                             }
{ TNS | TNSR | TNSE                                           }
{                                                             }
{ UNRESOLVED [, AXL ] | [, OCA ]                              }
{                                                             }
{ XREF                                                        }
{                                                             }
```

file-set

TACL-style file name pattern specifying a collection of object files. If *file-set* is not specified, the current subvolume is assumed.

AFTER *filename*

specifies the starting file for the operation. The first file is the one that alphabetically follows the specified filename.

START *filename*

specifies the starting file to display.

prep-info-option

used to analyze the prepared state of object files. See Preparing TNS Programs on page 3-3 for a description of the prepared state of a program.

DETAIL

displays an item-by-item analysis of each object file including:

- the type of the object file: TNS, TNS/R and so on.
- for SQL programs, whether the program is a valid SQL program, whether the program needs SQL compilation, and so on.
- the user library if there is one for this object.
- the prepared state of a program.
- the number of references to prepared external procedures.
- the object file attributes such as Inspect, Highrequesters and so on.
- for prepared programs, the NonStop AutoTMF software attributes configured for the program with the ADD ATMFPROGRAMS or ALTER ATMFPROGRAMS commands.

PREPARED

lists the object files that are prepared to use AutoTMF.

UNPREPARED

lists the object files that are not prepared to use AutoTMF.

Note: Programs that have been prepared but have some preparation conflict appear in both the PREPARED and UNPREPARED lists.
Moreover, PREPARED and UNPREPARED are "negative" options, each causing the other type of file to be omitted. Specifying both PREPARED and UNPREPARED displays only those that are prepared with a conflict.

AXLCHECK

displays all object files that are accelerated, have external references that are altered by a PREPARE command, and have code that make references to procedure labels.

COBOL

displays the COBOL and CRE information contained in an object file:

- a list of program units bound into the object file
- by program unit, a list of the files opened by the program, the type of file and the open parameters.

CONFLICTS

lists the procedures that duplicate intercepted procedures and external references to intercepted procedures that are in conflict.

PROC[EDURE]S [, OCA] [, AXL]

lists the procedures in each object file.

If AXL is specified, the procedures listed include the procedures invoked as a result of acceleration (millicode).

If OCA is specified, the procedures listed include information from the OCA-generated region in the object file.

PROG[RAMS]

limits the display of object file information to executable programs:

- TNS programs with a MAIN procedure
- TNS/R loadfiles that are not user libraries
- TNS/E loadfiles that are not DLLs

REBASE

generates an ELD (TNS/E linker) command that is used to copy and rebase a public DLL to the NonStop AutoTMF operational subvolume.

AutoTMF does not execute the command. The user executes the command prior to preparing a DLL.

The REBASE option is used by the AutoTMF INSTALL macro.

TNS | TNSR | TNSE

restricts the information displayed to the designated object file types: code 100 files for TNS, code 700 files for TNSR and code 800 files for TNSE.

UNRESOLVED [, OCA] [, AXL]

lists the unresolved external references for each object file.

If AXL is specified, the procedures listed include the procedures invoked as a result of acceleration (millicode).

If OCA is specified, the procedures listed include information from the OCA-generated region in the object file.

XREF

lists the retargeted procedure names that are referenced by the object file. The display indicates which names have been changed.

If *prep-info-option* or *misc-file-info* are not specified, the command displays one line of information for each file under headings: Program name, Type, SQL State (SQL programs only), User Library and Status.

LABELDISPLAY (LD)

Displays label information for the specified table or view, such as catalog name, audit compression, columns data types, primary and alternate key columns, partitions, allocated extents, and so on. The command can be abbreviated to LD.

```
L[ABEL]D[ISPLAY] sql-table [ ALLP[ARTS] | ALLI[NDEXES] ] ;
```

sql-table

a SQL table or View.

ALLP[ARTS]

displays the labels of all partitions of the table.

ALLI[NDEXES]

displays labels of all indexes of the table.

LISTFILEOPENS (LFO)

Displays all file opens like the FUP LISTOPENS command, with additional options. The command can be abbreviated to LFO.

```
L[IST]F[ILE]O[PENS] file-set [, options ];

options is:

{ BYF[ILE]                                                    }

{ BYO[PENER]                                                  }

{ OUTP[UT] | WRIT[E]                                          }
```

file-set

a TACL-style file name pattern specifying a collection of files. If *file-set* is omitted, all the opens for the local system are displayed ($* is assumed).

To display opens of temporary files, specify *file-set* as $*vol*.* or $*vol*.#*, where $*vol* can be “$*”.

BYF[ILE]

displays the opens by file name. It is the default.

BYO[PENER]

displays the opens by opener.

OUTP[UT] | WRIT[E]

displays files opened for write or update access only

LISTLOCKS (LL)

Displays granted and waiting lock requests for a file set. This command differs from FUP LISTLOCKS: NonStop AutoTMF software LISTLOCKS has a more compact display and more options and allows the user to show locks for the entire system with one command.

```
L[IST]L[OCKS] file-set [, option ];

option is:

{ BYF[ILE]                                                    }
{                                                             }
{ BYH[OLDER]                                                  }
{                                                             }
{ DEADL[OCKS] [, RESOLVE ]                                    }
{                                                             }
{ H[EX]                                                       }
{                                                             }
{ NOI[TENT]                                                   }
{                                                             }
{ SUM[MARY]                                                   }
{                                                             }
{ TXSTATUS                                                    }
{                                                             }
{ WAITING                                                     }
{                                                             }
{ WRAP                                                        }
{                                                             }
```

file-set

is a TACL-style file name pattern specifying a collection of Enscribe files. If this parameter is omitted, all the locks for the local system are displayed.

option

BYF[ILE]

display the locks by file name. This is the default.

BYH[OLDER]

display the locks by holder. Can be specified in conjunction with BYFile.

DEADL[OCKS] [, RESOLVE]

display sets of granted and waiting lock requests that form a deadlock. Deadlocks might resolve themselves if the waiting requests are using timed I/O.

Deadlocks usually involve multiple files on multiple volumes. Only deadlocks that are contained within the specified file-set are detected. Deadlocks that span multiple systems are not detected.

RESOLVE

allows the user to resolve the detected deadlock. The command displays a list of transactions and processes that are participants in the deadlock and prompts the user to select either a process to abend, or a transaction to abort to resolve the deadlock.

H[EX]

display the lock key is hexadecimal. This is useful for cases where the key fields are not ASCII.

NOI[TENT]

eliminates the display of intent locks. Intent locks are generally used to prevent another process or transaction from gaining a file lock while a record lock is held.

SUM[MARY]

displays only the count of record and file locks for each file lock-holder combination. This applies to BYFILE and BYHOLDER options.

TXSTATUS

displays a summary of all transactions or processes that are holding locks displayed by LISTLOCKS.

WAITING

display all the lock holders and waiters for files and/or records that have at least one wait request outstanding.

WRAP

display the entire key, using multiple lines if required.

LOG

Collects a history of the CI session to a file. LOG TO starts the logging process, and LOG STOP terminates the logging.

```
LOG { TO filename [CLEAR] | STOP };
```

TO *filename*

starts logging to *filename*.

filename may be a disk file, a printer, or another terminal. If the log file does not exist, the CI creates the log file as an edit file.

CLEAR

clears the log file of existing data.

STOP

closes the current log file and stops logging.

MODIFY AUTOTMF

Copies all entries of the tables that were defined for system *nodename* and creates the same entries for the local system. First move the ATMFFILE and ATMFPROG tables to the MAPDB and then issue the MODIFY AUTOTMF command.

If a configuration entry is already defined on the local system, that configuration entry is not copied from the original system.

```
MODIFY AUTOTMF REPLACE NODENAME nodename [ , LISTALL ];
```

nodename

the Expand node name of the original system.

LISTALL

displays the entries that are being modified in ATMFATTR and ATMFPROG.

Note. The command assists you in migrating the configuration of NonStop AutoTMF software file and program attributes to a new system. For details on migration procedures, see "Migrating the NonStop AutoTMF Software Configuration to a New System" on page A-7

MODIFY GLOBALS

Copies all the global parameters that were defined for system *n* and creates the same global parameters for the local system. If a global parameter is already defined for the local system, the parameter is not updated with the value that was configured on the original system *n*.

```
MODIFY GLOBALS REPLACE SYSTEMNUMBER n [, LISTALL ];
```

n

the original system number.

LISTALL

displays the value of the global parameters being modified.

Note. The command assists you in migrating the configured global parameters to a new system. For details on migration procedures, see "Migrating the NonStop AutoTMF Software Configuration to a New System" on page A-7

MODIFY MAPDBS

Copies all entries of MAPDBS that were defined for the system called *nodename* and creates the same entries for the local system. If the MAPDBS entry is already defined on the local system, the MAPDBS entry is not copied from the original system.

The command does not:

- copy the actual MAPDB tables
- verify that those tables are present on the local system
- make any changes that may be required to migrate the content of MAPDB tables to the local system.

```
MODIFY MAPDBS REPLACE NODENAME nodename [ , LISTALL ];
```

nodename

the Expand node name of the original system.

LISTALL

displays the entries that are being modified in the MAPDBS table.

Note. The command assists you in migrating the configured Monitor and MapDB attributes to a new system. For details on migration procedures, see "Migrating the NonStop AutoTMF Software Configuration to a New System" on page A-7.

MONITOR

The MONITOR command sends a command to the monitor process. Refer to Monitor Commands on page 6-91 for a list of valid monitor commands

```
MONITOR [ [\sys.]process-name | * ] command-text
```

[*sys.*]*process-name*

specifies an executing monitor process. If omitted, the monitor for the current MapDB will be used.

*

specifies all executing monitor processes.

command-text

is all text up to the end of the current command line. A MONITOR command cannot be continued on multiple lines. A semi-colon is neither required nor recognized as a command terminator.

NSKFIXUP

Operates on a collection of object files to alter the user library, causing the operating system to perform an object file fixup.

Prevents the “library conflict” error that can occur when multiple instances of the same program are started concurrently and an operating system fixup is required. The fixup is required when a program is compiled, bound, prepared, or moved, after a system cold load or when the user library is changed.

This command does not perform the functions of the PREPARE command. The file need not be prepared.

```
NSKFIXUP object-fileset, option;

option is:

{ AXCEL                                                    }
{                                                          }

{ LIBRARY library-file                                     }
{                                                          }
{ OCA                                                      }
{                                                          }
{ TNS | TNSR | TNSE                                        }
```

object-fileset

file set specifying a collection of object files.

option is*:*

AXCEL

instructs AutoTMF to accelerate the object file using AXCEL after the operating system fixup has completed.

This option allows the user to accelerate a collection of files through a single command.

LIBRARY *library-file*

name of the user library file to set in the object file-set. If omitted, the user library is not changed in the object file. If LIBRARY is specified without *library-name*, the current NonStop AutoTMF software runtime library will be used.

OCA

instructs AutoTMF to accelerate the object file using OCA after the operating system fixup has completed.

This option allows the user to accelerate a collection of files through a single command.

TNS | TNSR | TNSE

restricts the processing of the command to the designated object file types: code 100 files for TNS, code 700 files for TNSR, and code 800 files for TNSE.

Specifying the LIBRARY option implicitly restricts the processing of the command to the file type of the library object and is equivalent to specifying TNS, TNSR, or TNSE.

OBEY

Reads and runs a sequence of commands from another device. The commands are run serially until end-of-file is detected. An OBEY file may not contain an OBEY command.

```
OBEY filename;
```

filename

identifies the file containing a sequence of CI commands.

OPEN

Specifies the current NonStop AutoTMF software monitor process.

```
OPEN process-name;
```

process-name

specifies the name of a configured monitor process. The monitor process does not need to be running to update or display information for the corresponding MapDB.

OUT

Directs the output of the CI session to another file. Interactive terminal prompts will continue to appear on the original input device.

```
OUT [ filename ];
```

filename

identifies an output file and directs CI session output to that file. If filename is omitted, output will be directed to the original process OUT file.

PREPARE

Changes all external references to Enscribe and selected operating system procedures to the corresponding NonStop AutoTMF software runtime library procedures; also establishes the NonStop AutoTMF software runtime library as the user library for a TNS or TNSR program and as a DLL to a TNS/E program.

Only complete programs are prepared. Object files with no main procedure are either user libraries or components to be bound with other object files; these are skipped by the command.

```
PREPARE object-fileset [, option ];

option is:

{ !                                                                  }
{                                                                    }
{ AFTER filename                                                     }
{                                                                    }
{ ALLOWDUPLICATES                                                    }
{                                                                    }
{ LIBRARY alternate-library                                          }
{                                                                    }
{ OCA | NOOCA                                                        }
{                                                                    }
{ START filename                                                     }
{                                                                    }
{ TNS | TNSR | TNSE                                                  }
{                                                                    }
{ USERLIB | UL                                                       }
```

object-fileset

a file name pattern that describes a collection of application program files to be prepared.

option

!

forces the preparation of an object file that is normally not prepared; such files include ESCMON, ESCORT, ESCRUNTM, ESCRUNNT, ESCRUNNM, ESCZIP, FUP, PATHTCP2, and SORTPROG.

AFTER *filename*

specifies the starting file for the operation, which is the file that alphabetically follows *filename*.

ALLOWDUPLICATES

forces the preparation of an object file that contains a mixture of prepared and unprepared procedures and for which preparation will create duplicate external references. The object file must then be recreated in Binder to resolve the duplicates.

If ALLOWDUPLICATES is not specified and the CI detects this condition, the CI displays an error and the file is not prepared.

LIBRARY *alternate-library*

specifies an object file to set as the user library for the prepared object file. If the library does not contain the NonStop AutoTMF software runtime library, an error message is displayed and the object files are not prepared.

If this parameter in omitted, the user library is one of:

- The program's current user library which must be bound with the NonStop AutoTMF software runtime library. If the program is not bound with the NonStop AutoTMF software runtime library, an error message is displayed and the object file is not prepared.
- The NonStop AutoTMF software runtime library that is used by the CI.

OCA | NOOCA

specifies whether a TNS object (file code 100) is accelerated using OCA after the object has been prepared.

By default, PREPARE automatically processes a TNS object using OCA whenever PREPARE invalidates the OCA region. If the program has no OCA region, no extra processing is performed. This option overrides the default behavior.

By specifying NOOCA, PREPARE never processes a program using OCA. If the preparation invalidates the OCA region, the program runs in interpreted mode, unless you manually process the program using OCA.

START *filename*

specifies the starting file to prepare *fileset.*

TNS | TNSR | TNSE

restricts the processing of the command to the designated object file types: code 100 files for TNS, code 700 files for TNSR, and code 800 files for TNSE.

By default, PREPARE processes only TNS/R objects on NonStop S-Series servers and only TNS/E objects on Integrity NonStop servers. It processes TNS objects on all servers. Specifying this option overrides the default behavior.

Specifying the LIBRARY option implicitly restricts the processing of the command to the file type of the library object and is equivalent to specifying TNS, TNSR, or TNSE.

USERLIB | UL

prepares of object files with no main procedure.

If USERLIB is not specified, the CI displays an error and the file is not prepared.

Prepare any application user libraries prior to preparing the application programs. See Combining a User Library with the NonStop AutoTMF Runtime on page 3-8 for details

about how to combine the NonStop AutoTMF software runtime library with a user library.

PREPARE preserves existing references to user libraries and checks that the references are properly prepared before preparing the programs.

Note. Preparing a licensed privileged program removes the license attribute of the object file. You must re-license the program after it has been prepared.

PROGINFO (PI)

Displays information about object files. If NonStop AutoTMF software program attributes are configured for an object file in the file set specified, these attributes are also displayed. This information is also displayed by the INFO PROGRAM command.

```
P[ROG]I[NFO] [file-set ] [, AFTER filename | START filename ]
             [,prep-info-option | object-info-option ];

or PID file-set;

prep-info-option is

{ DETAIL                                                      }
{                                                             }
{ PREPARED                                                    }
{                                                             }
{ UNPREPARED                                                  }

object-info-option is

{ AXCELCHECK                                                  }
{                                                             }
{ COBOL                                                       }
{                                                             }
{ CONFLICTS                                                   }
{                                                             }
{ PROCS | PROCEDURES [, AXL ] | [, OCA ]                      }
{                                                             }
{ PROG[RAMS]                                                  }
{                                                             }
{ REBASE                                                      }
{                                                             }
{ TNS | TNSR | TNSE                                           }
{                                                             }
{ UNRESOLVED [, AXL ] | [, OCA ]                              }
{                                                             }
{ XREF                                                        }
```

file-set

TACL-style file name pattern specifying a collection of object files. If *file-set* is not specified, the current subvolume is assumed.

AFTER *filename*

specifies the starting file to display, which is the first file that alphabetically follows the specified filename.

.START *filename*

specifies the starting file to display.

For a description of *prep-info-option* and *object-info-option*, see INFO PROGRAM entries.

PID *file-set*

Abbreviates PROGRAM INFO *file-set* DETAIL;

PURGEDATA

Clears data from a file.

```
PURGEDATA file-name;
```

file-name

the name of a file.

RESET

Changes all Escort CI environmental variables to their original settings. Environmental variables are variables such as the process completion code.

```
RESET;
```

RESET GLOBAL

Resets the value of a global parameter to its original default.

```
RESET GLOBAL [ global-parameter | * ];
```

global-parameter

parameter to reset. For a list of global parameters, see command description for ALTER GLOBAL on page 6-34.

*

resets all global parameters to their default value.

RESET LOCAL

Resets the value of a local parameter to its default value, which is the value of the corresponding global parameter.

```
RESET GLOBAL [ local-parameter | * ];
```

local-parameter

parameter to reset.

Local parameters are the same as global parameters. For a list of local and global parameters, see command description for ALTER GLOBAL on page 6-34.

*

resets all local parameters to their default (global) value.

RUN[D]

Runs a program during a CI session. When the program terminates, the session resumes.

```
RUN[D] filename[/run-options/] [command]
```

filename

specifies a program file to be run.

run-options

specifies standard TACL process options, including the following:

CPU	EXTSWAP	LIB	NOWAIT	PRI
DEBUG	IN	MEM	OUT	SWAP
DEFMODE	INSPECT	NAME	PFS	TERM

command

specifies any command line to be passed to the process in the startup message.

The RUN[D] command is not a multiline command and is not terminated with a semicolon. Any semicolon is passed to the process as part of the startup command.

Refer to Table 6-3 for a list of standard programs, such as EDIT and SQLCI, that can be run by simply typing the program name. These commands are equivalent to specifying “RUN $SYSTEM.SYSTEM.” before the name and otherwise follow the complete RUN command syntax.

START MONITOR

Starts a NonStop AutoTMF software monitor process for one MapDB or all MapDBs.

```
START MONITOR [ process | mapdb ] [, option ];

option is:

{ PRIMARY cpu-number                                                }
{                                                                   }
{ BACKUP cpu-number                                                 }
{                                                                   }
{ PRIORITY process-priority                                         }
{                                                                   }
{ HOMETERM file-name                                                }
{                                                                   }
```

process

specifies the name of the NonStop AutoTMF software monitor process to be started.

mapdb

specifies the subvolume of a MapDB. This is an alternative method to specify the process to be started.

If the parameter is omitted, the default monitor is assumed, most commonly $ZESC. Use the ENV command to display the current default monitor and MapDB.

option

PRIMARY *cpu-number*

cpu (in the range of 0 to 15) of the primary monitor processes. Used when the monitor process is started.

BACKUP *cpu-number*

cpu (in the range of 0 to 15) of the backup monitor processes. Used when the monitor process is started.

PRIORITY *process-priority*

specifies the priority (in the range of 1 to 199) for the monitor process. Used when the monitor process is started.

HOMETERM *file-name*

specifies a home terminal for the monitor process. Used when the monitor process is started.

If omitted, the home terminal is set to the default EMS collector. The default EMS collector is $0, or the EMS collector configured by setting the global parameter EMSCOLLECTOR.

Note that internally, the default hometerm of the monitor is $ZHOME, as displayed by the TACL command STATUS $<*process-name*>, because an EMS collector cannot be specified as the hometerm of a process when the process is launched. Functionally however, the home terminal for the monitor process is the default EMS collector, as shown by the AutoTMF STATUS MONITOR command.

STATS

Scans an Enscribe file to determine the record count and distribution of record lengths. STATS is useful to determine if file records conform to expected lengths.

```
STATS file-name;
```

file-name

is an Enscribe file name.

STATUS MONITOR

Determines the status of the monitor process by sending the monitor inquiries.

```
STATUS MONITOR [ * | [\sys.]process-name | mapdb ];
```

*

queries all configured monitor processes.

[*sys.*]*process-name*

specifies the name of the NonStop AutoTMF software monitor process to be queried. The default is the current monitor for the session.

mapdb

specifies the subvolume of a MapDB. This is an alternative method to specify the process to be queried.

STATUS TRACE

Request a status of the current active traces from the monitor process. This command replaces the MONITOR STATUS TRACE command.

```
STATUS TRACE trace-index;
```

trace-index

a number from 0 to 7 that specifies the index number of the trace to report on.

If *trace-index* is omitted, the status of all active traces is displayed.

STOP MONITOR

Performs an orderly shutdown of the monitor process.

```
STOP MONITOR [ process-name | mapdb ];
```

process-name

specifies the name of the NonStop AutoTMF software monitor process to be stopped. The default is the current monitor for the session.

mapdb

specifies the subvolume of a MapDB. This is an alternative method to specify the process to be stopped.

STOP PROCESS

This command attempts to stop a process in a way that allows NonStop AutoTMF software to commit all active automatic transactions. To stop a process, the command must be issued by the user ID or the manager ID of the process creator, or by the SUPER user.

ESCORT CI sends a special message to the process, so the process must be reading $RECEIVE.

This command is intended to be used in emergencies where automatic transactions are not being committed and you want to prevent transaction aborts and loss of data. When the message is received, the automatic transactions are committed and the process is abended. A message is written to the process home terminal and to the EMS log.

```
STOP PROCESS { cpu,pin | process-name };
```

cpu,pin | *process-name*

specifies the process id or name of the application process to be stopped.

TIME

Displays the current local time of day as obtained from the local Guardian timekeeping function. You may also request the local time as seen by another system on your Expand network.

```
TIME [ systemname ];
```

systemname

identifies an Expand system for which the local system time is desired.

TRACE

Specifies a NonStop AutoTMF software trace configuration and enables tracing. When prepared programs begin execution, the programs access the trace configuration and, if enabled, begin tracing selected operations to the trace file.

Programs already executing are not traced unless dynamic tracing is enabled (see ALTER GLOBAL).

Tracing is a diagnostic tool for NonStop AutoTMF software problems. The TRACE command should be used with caution in a production environment.

Note that the user should not rely on the current format of the trace output remaining the same over time. The output format of the trace is subject to change.

```
TRACE [trace-index] [traced-entity ]
                            TO {filename | MYTERM }
                            [, process-specs]
                            [, option];
or

TRACE STOP;

traced-entity is

{ EXEC[UTION]                                                    }
{                                                                }
{ KEYS                                                           }
{                                                                }
{ DATA                                                           }

process-specs is

{ PROGRAM { object-file-name | fileset }                         }
{                                                                }
{ PROCESS { process-name | process-namelist | * }                }
{                                                                }
{ PROCESS { cpu,pin }                                            }
{                                                                }
{ PROCESS { cpu,* }                                              }

option is

{ CLEAR                                                          }
{                                                                }
{ DEBUG { TERM term-name | MYTERM }                              }
{                                                                }
{ EDIT                                                           }
{                                                                }
{ FOR n                                                          }
{                                                                }
{SEPARATE                                                        }
{                                                                }
{TRANSID                                                         }
```

filename | MYTERM

name of the trace file. If MYTERM is specified, the trace output is sent to the home terminal of the CI that issued the TRACE command.

The trace file is opened when the process is started. The file must be a process or an entry-sequenced file with a record length of at least 132 bytes. If the file does not exist, the file is created.

The trace file can be an Edit file. Edit files provide for the most efficient tracing, but an Edit trace file can be opened by only one process at a time.

trace-index

The runtime allows for 32 concurrent active traces. This value from 0 to 31 assigns an index to the tracing operation. If the option is omitted, the runtime assumes *trace-index* 0.

When a user is allocated a trace index, the user owns the entry until the trace expires or is stopped. Only that user (or super) can stop the trace.

traced-entity

EXEC[UTION]

Trace the following information:

- Enscribe calls such as KEYPOSITION, READ, WRITE intercepted by the NonStop AutoTMF software runtime library.
- The startup message.
- PARAMS and ASSIGNs passed to the process.
- DEFINEs active when the process started.

Each entry is timestamped and contains the name (or CPU, PIN) of the calling process.

KEYS

In addition to EXECUTION information, trace key fields from read, write and update operations. This option generates less output than DATA; key information is sufficient to diagnose most program logic problems.

When activating a KEY or DATA trace with this command, the following conditions and restrictions apply:

- The trace file must be a disk file.
- The process access id and creator access id of the tracing process must be the same.
- The trace file must be owned by the process access id of the tracing process. The security on the trace file must be set to “ONOO”.

- When a security violation occurs, tracing is turned off.

These rules guarantee that unauthorized users cannot gain access to data.

DATA

In addition to KEYS, trace entire data records sent to or returned from read, write and update operations.

process-spec

PROGRAM *object-file-name* | *fileset*

restrict tracing to a selected set of programs.

If this option is used in conjunction with the DEBUG option, the program must be uniquely specified, not as a pattern.

PROCESS *process-name* | *process-namelis*t | *
PROCESS *cpu-pin*
PROCESS *cpu*,*

restrict tracing to a selected set of processes.

If PROCESS is used with the DEBUG option, the process must be uniquely specified (not as a pattern).

If PROGRAM or PROCESS are not specified, all prepared programs are traced.

option

CLEAR

purges the data in the trace file before writing the first trace entry.

DEBUG [TERM *term-name* | MYTERM]

starts the traced process in INSPECT and sends the INSPECT prompt to one of the following locations:

- to the terminal specified in the TERM option or,
- if MYTERM is specified, to the home terminal of the CI session or,
- if none is specified, to the home terminal of the process being debugged.

The DEBUG option can be specified only when tracing a single process; a unique PROGRAM or PROCESS must be specified.

This option cannot be used for dynamic tracing.

EDIT

instructs NonStop AutoTMF software to create the trace file as an Edit file. If this option is specified, only one process can open the trace file at a time.

FOR *n*

sets a time limit of *n* minutes for tracing, starting when the command is entered. The default is 60 minutes. If *n* is set to 0, then there is no time limit. When the time limit expires, the trace configuration is reset and processes currently tracing with the configuration cease tracing.

SEPARATE

specifies that each process trace should be written to a separate trace file. Otherwise, all traces for a specific trace definition are written to the file specified in the command.

When specifying SEPARATE, the trace file name must not have more than five characters, as a three-digit sequence number is appended to the file name to create unique file names (TRACE000, TRACE001, and so on).

The collection of trace files is managed by the user. Each trace activation attempts to find some unused name of the form [file]*nnn*. If all 1000 file names are in use, the trace is not performed.

TRAN[SID]

displays the current process transaction identifier after every intercepted operation.

TRANSID generates many additional trace entries and is provided for the purpose of diagnosing specific AutoTMF problems. Use it only if instructed to do so by support.

By default, trace entries for BEGINTRANSACTION, RESUMETRANSACTION and ENDTRANSACTION operations show the current transaction whenever it changes, which provides sufficient information for most diagnostic needs.

STOP

resets the trace configuration. New processes are not traced, but processes currently tracing continue to trace until the time limit expires.

UNPREPARE

The UNPREPARE command restores a prepared object file to its original state. UNPREPARE reverses the effects of the PREPARE command.

```
UNPREPARE object-fileset [option ];

option is:

{ AFTER filename                                                    }
{                                                                   }
{ ALLOWDUPLICATES                                                   }
{                                                                   }
{ LIBRARY alternate-library | NONE                                  }
{                                                                   }
{ OCA | NOOCA                                                       }
{                                                                   }
{ START filename                                                    }
{                                                                   }
{ TNS | TNSR | TNSE                                                 }
```

object-fileset

a file pattern that describes a collection of application program files to be prepared.

option

AFTER *filename*

specifies the starting file for the operation. The first file unprepared is the one that alphabetically follows the specified filename.

ALLOWDUPLICATES

forces the unprepare of an object file that contains a mixture of prepared and unprepared procedures and for which this operation will create duplicate external references. The object file must then be recreated in Binder to resolve the duplicates.

Duplicate conditions also occur when UNPREPARE is issued on a prepared user library that has been bound with the NonStop AutoTMF software runtime library. To unprepare a user library, do not use the UNPREPARE command; replace the user library with its original unprepared version.

If ALLOWDUPLICATES is not specified and the CI detects this condition, the CI displays an error and the file is not unprepared

LIBRARY *alternate-library* | NONE

specifies a file to set as the user library for the object files in the fileset.

If NONE is specified, the library pointer is removed from the object file.

If this option is not specified, the library is set as follows:

- If a program points to one of ESCRUNTM, ESCRUNN, ESCRUNNT, ESCRUNNM or ESCRUNNL, the library pointer is removed from the object file.
- If the program points to a library other than ESCRUNTM, ESCRUNN, ESCRUNNT, ESCRUNNM, or ESCRUNNL, ESCORT CI checks that the NonStop AutoTMF software library is no longer bound into the user library. If the NonStop AutoTMF software library is bound into the user library, however, an error is displayed and the object file is not unprepared.
- If the program is a TNS/E program using ESCRUNDL, ESCRUNDL is no longer used as a DLL to the program.

OCA | NOOCA

specifies whether a TNS object (file code 100) is accelerated using OCA after the object has been unprepared.

By default, UNPREPARE automatically processes a TNS object using OCA whenever UNPREPARE invalidates the OCA region. If the program has no OCA region, no extra processing is performed. This option overrides the default behavior.

By specifying NOOCA, UNPREPARE never processes a program using OCA. If the preparation invalidates the OCA region, the program runs in interpreted mode, unless you manually process the program using OCA.

START *filename*

specifies the starting file to unprepare in *fileset*.

name, the current NonStop AutoTMF software runtime library will be used.

TNS | TNSR | TNSE

restricts the processing of the command to the designated object file types: code 100 files for TNS, code 700 files for TNSR and code 800 files for TNSE.

By default, UNPREPARE processes only TNS/R objects on NonStop S-Series servers and only TNS/E objects on Integrity NonStop servers. It processes TNS objects on all servers. Specifying this option overrides the default behavior.

Specifying the LIBRARY option implicitly restricts the processing of the command to the file type of the library object and is equivalent to specifying TNS, TNSR, or TNSE.

Note. Unpreparing a licensed privileged program removes the license attribute of the object file. You must re-license the program after it has been unprepared.

UPDATE

Modifies records in an Enscribe file. The syntax for specifying records to update is the same as the COPY command.

The command displays the record in the format that is specified and allows the user to modify the record values.

```
UPDATE file [, update-options ];

update-options is:

{ selection-option          }
{ display-option            }

selection-option is:

{, COMP[ARELEN] n                                                  }
{                                                                  }
{, COUNT num-records                                               }
{                                                                  }
{, EXACT                                                           }
{                                                                  }
{, FIRST { ordinal-record-num |                                    }
{{ KEY {record-spec|key-value [, key-value ]}|                     }
{{ key-spec ALTKEY key-value [, key-value ] } }                    }
{                                                                  }
{, FROMLAST                                                        }
{                                                                  }
{, REVERSE                                                         }
{                                                                  }
{, SHARE                                                           }
{                                                                  }
{, SKIPMATCH                                                       }

display-option

{, [O]CTAL                                                         }
{                                                                  }
{, [D]ECIMAL                                                       }
{                                                                  }
{, [H]EX                                                           }
{                                                                  }
{, [A]SCII                                                         }
{                                                                  }
{, BYTE                                                            }
{                                                                  }
{, NO HEAD                                                         }
```

file

is the name of the Enscribe file that is to be modified by the command.

selection-options

COMP[RELEN] *n*

sets the compare length for generic positioning on the record key (primary or alternate) specified in the FIRST KEY option. *n* is between 1 and 255 and must be less than or equal to the key specified.

COUNT *num-records*

is the number of records or rows to be copied. If omitted, all records are copied

EXACT

requests exact positioning on the record key (primary or alternate) specified in the FIRST KEY option.

FIRST { *ordinal-record-num* }
{ KEY { *record-spec* | *key-value* } }
{ *key-spec* ALTKEY *key-value* [, *key-value*] }

the starting record of the input file to update. If omitted, the update starts at the first record or row in the input file.

ordinal-record-num

the number of records or rows from the beginning of the file that are to be skipped. The first record in a file is record zero. The maximum value is 4294967295.

KEY { *record-spec* | *key-value* }

the primary key value for the starting record or row of a structured disk file.

record-spec is a fixed quantity limited to 2**63 -1.

- for unstructured files, *record-spec* is the starting relative byte address
- for relative files, *record-spec* is the starting record number
- for entry-sequenced files, *record-spec* is the 64-bit numeric record key

key-value applies only to key-sequenced files and specifies the approximate position of the starting record; *key-value i*s specified as a string or as integer byte values in the range of 0 to 255.

The key-value is entered as follows:

"[" { string } [, { string }] "]"

{ 0:255 } [{ 0:255 }]

where the integers represent the byte values. To specify a list of strings, enclose each string in quotation marks separated by a comma, and enclose the list in square brackets.

key-spec

the alternate key tag (a 2-byte string or a 16-bit integer) designating the alternate key to be used for positioning.

ALTKEY *key-value* [, *key-value*]

the alternate key of the starting record or row. The format of *key-value* is described above.for entry-sequenced files, *record-spec* is the *ordinal-record-number*.

FROMLAST

positions on the last record in the key range specified in the FIRST KEY option.

REVERSE

reads the input file from the starting record in reverse order.

SHARE

opens the file in share exclusion mode. The default is protected.

SKIPMATCH

positions in the input file to the record immediately following the one whose key matches the key specified in the FIRST KEY option. The entire key must be supplied. If the file is not of type key sequenced, an error 46 is returned.

This option applies to key sequenced files only.

display-options

Display options differ slightly from FUP display options: output data transformations (RECOUT, BLOCKOUT, PAD and so on) are applied before formatting the data. Thus, formatted data is displayed as the data would be written to a disk file.

Entry-sequenced record keys are displayed in the 64-bit format

[O]CTAL

displays the output in octal and ASCII format.

[D]ECIMAL

displays the output in decimal and ASCII format

[H]EX

displays the output in hexadecimal and ASCII format

[A]SCII

displays the output in ASCII format. This option is ignored if combined with OCTAL, HEX, DECIMAL or BYTE display options.

NO HEAD

omits the heading preceding each record or row displayed, when one of the *display-options* is specified.

UPGRADE MAPDB

Creates additional tables in an existing MapDB. The tables are created if required for a new release of NonStop AutoTMF software. The CI prompts you to issue the command when necessary.

```
UPGRADE MAPDB subvol;
```

subvol

the name of the MapDB subvolume to upgrade.

VOLUME

Changes the default volume and/or subvolume for filename expansion. VOLUME can be abbreviated to V.

```
V[OLUME] [ volume | subvolume | volume.subvolume ];
```

volume

specifies a new default volume; does not alter the default subvolume.

subvolume

specifies a new default subvolume; does not alter the default volume.

Monitor Commands

Each of the following commands are recognized by the monitor process. To send a command to the monitor, use the MONITOR command of the CI.

LOG

Initiates or terminates a monitor activity log.

```
MONITOR LOG { TO file [CLEAR] | STOP }
```

TO *file* [CLEAR]

specifies the name of a log file and starts logging. CLEAR empties the file before logging starts.

If *file* already exists, information is appended to the end of the file unless CLEAR is specified. If *file* ends with a number (for example LOG001), additional files will be allocated with incrementing numbers when the first file becomes full.

STOP

terminates logging.

Activities that are logged include:

- Starting and stopping the log.
- Any errors obtaining configuration information from MapDB.
- Monitor opens and closes by requesting processes.
- The result of ATMFFILESET and ATMFPROGRAM configuration requests.
- Any mismatch between an NonStop AutoTMF software runtime library and the monitor or MapDB.
- Fault tolerance process events, such as a backup process takeover.
- Local CPU failures and reloads.
- Remote network status changes

Note. Information in the activity log is for informational and diagnostic purpose. It is not intended to be an official or reliable record for continuing management or control purposes. The content and format of the log is subject to change without notification.

STATUS

Displays information about various objects.

```
MONITOR STATUS { PROCESS | LOG  | LICENSE | TRACE   }
```

PROCESS

requests information about the monitor process.

LOG

requests information about the monitor log file, if any.

LICENSE

requests information about the NonStop AutoTMF software license for the current system.

TRACE

requests information about the currently specified trace, if any. This command has been replaced by the ESCORT CI command STATUS TRACE.

SECURITY

Instructs the monitor to accept further commands only from a specific set of process access ids.

```
MONITOR SECURITY { N | C | U | A | G | O | - }
```

The command option is a single character and is a Guardian security specifier. The command requestor must match the monitor process access id at the specified security level.

BACKUPCPU

The BACKUPCPU command specifies the cpu for the monitor backup process.

```
MONITOR BACKUPCPU [ cpu ]
```

cpu

specifies the cpu number where the monitor backup process should be started. If omitted, the backup process is terminated.

SWITCH

The SWITCH command causes the monitor primary and backup processes to exchange roles.

```
MONITOR SWITCH
```

A System Management

System Database

The System Database (SysDB) consists of two SQL tables:

- MAPDBS to store the description of map databases and associated monitor processes.
- REGISTRY to register a product license and other system-wide default configuration parameters referred to as globals.

Normally, each system has a single SysDB in the $SYSTEM.ESCSYSDB subvolume, but SysDB may also be stored on another volume. There should be only one SysDB subvolume named ESCSYSDB.

You need write access to the SysDB to create or alter the map database, to specify the product license or alter the default global parameters. You need read access to run some commands, especially to start the monitor process. Application processes do not access the SysDB, since application processes obtain all information through the monitor process.

When NonStop AutoTMF software needs to access the SysDB, NonStop AutoTMF software looks first for SysDB tables in $SYSTEM.ESCSYSDB. If the tables do not exist, NonStop AutoTMF software then searches all disks for a SysDB in the ESCSYSDB subvolume. If there is more than one SysDB, the command interpreter will terminate with an error.

Although unusual, you can configure more than one SysDB to define different NonStop AutoTMF software environments on the same system. This is most commonly required on development systems where different groups with very different requirements share a system. See Configuring an Alternate SysDB on page A-7 below for details.

Map Database

A Map Database (MapDB) contains NonStop AutoTMF software configuration information for selected files and programs. In this release, the MapDB subvolume contains three SQL tables:

- ATMFATTR to store NonStop AutoTMF software file configuration.
- ATMFPROG to store NonStop AutoTMF software program configuration.
- REGISTRY to store NonStop AutoTMF default configuration parameter values referred to as locals. Locals are analogous to globals, but apply to one MapDB only.

Each MapDB table must be secured for read and write access by the users who run NonStop AutoTMF software configuration commands or who start the monitor process. Application processes do not access the MapDB.

NonStop AutoTMF software supports the creation of an alternate MapDB, primarily to provide separation between production and development activities. Use of an alternate MapDB requires a DEFINE for each process execution and is not recommended for production applications. See section Configuring an Alternate MapDB on page A-5 for details on configuring an alternate NonStop AutoTMF software environments.

Monitor Process

The monitor process provides NonStop AutoTMF software configuration information for application processes; the monitor must be running to use NonStop AutoTMF services. Initially, the monitor is started when the MapDB is created. The monitor is a fault-tolerant process pair, so the monitor should not require restarting unless the monitor is stopped or a system coldload is performed.

As described in Update System Coldload Procedures on page 2-9, the system coldload procedures should start the monitor process. The monitor should be started after TMF is up and before starting Pathway or the application programs.

The monitor process is quite efficient and can service many hundreds of requests per second. The monitor should be run at high priority, since the monitor serves all NonStop AutoTMF applications and does not run for long periods.

Configuring the Monitor Process

The monitor is configured and started when the MapDB is created with the CREATE MAPDB command. The monitor has an assigned process name, $ZESC, reserved for NonStop AutoTMF software.

The monitor is configured with the ALTER MONITOR command. You can configure process priority, process CPUs, and the monitor event logging device.

Configuring Monitor Priority

All prepared programs will communicate with the monitor when a process is started. The monitor supplies all global configuration information and, if specified, any configuration for the program name.

Further, all prepared programs will communicate with the monitor if an audited file is opened and there is some NonStop AutoTMF software configuration for specific files.

The monitor caches all configuration information in memory and has efficient lookup algorithms; thus, each communication is usually a simple message and requires little CPU time and no disk I/O. The monitor can process a great number of information requests without imposing a significant load on the system.

You must set the monitor process priority high enough to avoid delays due to any process that may be CPU-bound. If the monitor is delayed, many other application processes will also be stalled when they start up or open audited files.

To avoid this situation, If the monitor receives a request from a process running at a higher priority, it automatically raises its own priority to the priority of the requesting process plus one.

You should configure the monitor priority to be above all application processes. A typical priority would be the same as a production PATHMON process. A typical command to configure the monitor priority would be:

```
ALTER MONITOR, PRIORITY 180;
```

Starting a Monitor

If a monitor is not running, start the monitor by using the START MONITOR command.

On the START MONITOR command, you can specify the monitor configuration options, but you should also alter the monitor configuration to ensure those options will be used if the monitor is restarted.

The CREATE MAPDB, ALTER MAPDB or ALTER MONITOR commands specify or change the configured values for options to be used by a START MONITOR command. For these to take effect, you must stop and restart the monitor.

Stopping a Monitor

The monitor process must be available whenever application programs are executing. If the monitor is stopped, executing processes can continue to access files that are already opened, but this situation should be avoided.

In general, processes executing when the monitor is stopped will operate as if NonStop AutoTMF software were not installed. No automatic transaction management will occur. If a program attempts to access an audited file without a transaction, it will typically encounter an error and fail.

In normal situations, the monitor process should be stopped only after all programs that access audited files have been stopped.

The STOP MONITOR command should be used to stop the monitor.

Configuring and Using an Alternate NonStop AutoTMF Environment

You may configure an alternate NonStop AutoTMF software environment for the following reasons:

- If you need to test or run different versions of NonStop AutoTMF software on the same system.
- If the different groups or applications require different NonStop AutoTMF file and program attributes.

To create an alternate environment, you must configure an alternate monitor and MapDB, as described below in Configuring an Alternate SysDB. You can also define default global values that apply to this alternate environment by configuring LOCAL parameters.

You do not need to configure separate NonStop AutoTMF software environments for performance reasons. Unlike other subsystems, there are no significant limits to the number of programs, users, database files, and so on, that can be handled efficiently with a single NonStop AutoTMF software environment.

If you configure the alternate environment to test or run a different version of NonStop AutoTMF, you also must install the alternate version of AutoTMF software in a separate subvolume as described below in Installing an Alternate Version of NonStop AutoTMF Software

If the environments need to be completely segregated, you may also configure a separate SysDB and MapDB. See Configuring an Alternate SysDB below.

Installing an Alternate Version of NonStop AutoTMF Software

If you wish to run different versions of NonStop AutoTMF software on your system, you first must install the alternate version in its own product subvolume. Run the INSTALL macro from the installation subvolume that contains the desired version, specifying the

alternate product subvolume of your choice. In the example below, the user creates a product subvolume $DATA.ATMFAAO to test NonStop AutoTMF SPR labelled AAO

```
1> v y0581aao
2> INSTALL $data.atmfaao
HP Nonstop(tm) AutoTMF(tm) Software File Installer

UNPAK - File decompression program - T1255H01 - (2007-03-21)

Archive version: 1
File Mode RESTORE Program - T9074H01 (20SEP2006) (AFJ)
(C)2000 Compaq (C)2006 Hewlett-Packard Development Company, L.P.
Drives: (\ATOM.$Y3R5)
System: \ATOM  Operating System: H06  Tape Version: 3
Backup options:  NO AUDITED, BLOCKSIZE 8, NO IGNORE, NO OPEN, PARTONLY OFF,
                 INDEXES IMPLICIT
*WARNING-7147*  Files created and stored via OSS and SQL/MX objects are not
                 supported.
Restore time: 26Mar2008  8:51  Backup time: 21Mar2008  0:49           Page: 1

Tape: 1       Code             EOF      Last modif   Owner RWEP   Type Rec  Bl

$DATA.ATMFAAO
 ESCERROR                    258048 21Mar2008  0:22 170,46  NUNU   K    128  4

 ESCFLTR       845              114 13Mar2008  9:57 170,46  NUNU
 ESCHELP                    1478656 17Mar2008 17:50 170,46  NUNU   K   1928  4

 ESCMON        100           6178816 21Mar2008  0:49 170,46  NUNU
 ESCORT        100           5603206 21Mar2008  0:49 170,46  NUNU
 ESCRUNDL      800          10246080 21Mar2008  0:32 170,46  NUNU
 ESCRUNN       700           3121792 21Mar2008  0:32 170,46  NUNU
 ESCRUNNL      700           5742344 21Mar2008  0:32 170,46  NUNU
 ESCRUNNM      700           6049800 21Mar2008  0:32 170,46  NUNU
 ESCRUNNT      700           3251496 21Mar2008  0:32 170,46  NUNU
 ESCRUNTM      100          12797952 21Mar2008  0:46 170,46  NUNU
 ZESCTMPL      839             57344 21Mar2008  0:25 170,46  NUNU   K    510  4


Summary Information

Files restored = 12  Files not restored = 0
Rebase and Prepare Public DLLs...
$system.ZDLL008.ZOSSKDLL rebased to $data.atmfaao.ZOSSKDLL and prepared
$system.ZDLL008.ZICNVDLL rebased to $data.atmfaao.ZICNVDLL and prepared
$system.ZDLL008.ZI18NDLL rebased to $data.atmfaao.ZI18NDLL and prepared
$system.ZDLL008.ZCRTLDLL rebased to $data.atmfaao.ZCRTLDLL and prepared
$system.ZDLL008.ZCREDLL rebased to $data.atmfaao.ZCREDLL and prepared
$system.ZDLL008.ZCOBDLL rebased to $data.atmfaao.ZCOBDLL and prepared

AutoTMF software file installation is complete.
Proceed to installation instructions described
in the user manual.
```

Configuring an Alternate MapDB

To configure an alternate MapDB and Monitor for your programs, you must perform the following:

1. Create an alternate MapDB
2. Associate the alternate MapDB and monitor with programs

Create an alternate MapDB

Use the CREATE MAPDB command specifying the alternate monitor process name.

For example, to create a test MapDB that is called TSTMAPDB and uses the monitor $ZTST:

```
AutoTMF 1? create mapdb tstmapdb,monitor $ztst;
--- MapDB table $PRPC.TSTMAPDB.REGISTRY created.
--- MapDB table $PRPC.TSTMAPDB.ATMFATTR created.
--- MapDB table $PRPC.TSTMAPDB.ATMFPROG created.
--- MapDB for AutoTMF created.
--- SysDB updated
--- Changes will take effect when the Monitor is restarted.
--- Starting Monitor process $ZTST

AutoTMF Monitor 1.8.1 - 23APR2007 -- $ZTST (5,719) - System \NSK
Started at May 8 2007  16:04:12, elapsed time = 0:00:00
Backup process $Ztst (4,785), no takeovers

Monitor hometerm: $0
```

Associate the alternate MapDB and monitor with programs

There are 3 ways to tell programs (via the Escort runtime) to use an alternate monitor:

- Create an edit file called ESCMONAM containing the name of the alternate monitor. The file must reside in the subvolume where the NonStop AutoTMF library resides. If programs have a user library that contains the escort runtime, ESCMONAM must reside in the subvolume where your user library resides. This is the simplest method to specify an alternate monitor process name. However, note that, if the =_ESCORT_MONITOR define has been added (method described below), it overrides the monitor name specified in the ESCMONAM file.
- Add the =_ESCORT_MONITOR define prior to executing prepared programs that will use the alternate monitor process. The monitor name specified with this define overrides the monitor name specified in the ESCMONAM file.
- Add a special procedure into the Escort runtime to hard-code the name of the monitor into the runtime. This method is considered a “legacy” method, as it can only be used with TNS and TNS/R programs. Moreover, the code change must be performed every time a new version of Escort is installed. See Hard-Coding Monitor Name into the Runtime on page A-11 below for details.

For escort, the NonStop AutoTMF command interpreter, there are 2 additional ways to use an alternate monitor process:

- Specify the monitor process name on the run command.

```
$DATA SUBVOL 7> escort $ztst
HP Nonstop(tm) AutoTMF(tm) Command Interpreter(T0581H01) - System \NSK
(C)2006 Hewlett Packard Development Company, L.P.
(C)2006 Carr Scott Software Incorporated
```

- Issue the OPEN command after executing the CI:

```
AutoTMF 2? open $ztst;
```

Configuring an Alternate SysDB

The SysDB has been engineered to be version-tolerant; multiple versions of NonStop AutoTMF software can use the same SysDB concurrently. If however, you must completely segregate operating environments, you can also configure a separate SysDB and MapDB.

To configure an alternate system database, the following steps are required:

1. Use the =_ESCORT_SYSDB DEFINE to specify an alternate SysDB subvolume. Do not use the ESCSYSDB subvolume name.
2. In the CI, issue the CREATE SYSDB command to create the new SysDB.
3. Enter your license.
4. Issue the CREATE MAPDB command to create an alternate MapDB. You must specify an alternate monitor process name.
5. Use the =_ESCORT_SYSDB DEFINE whenever using the CI. This is especially important when issuing the START MONITOR command. The alternate SysDB DEFINE is not required to run application programs, since the application programs do not access SysDB, except through the monitor process.
6. Associate the new MapDB and monitor with programs by using one of the methods described above in Associate the alternate MapDB and monitor with programs.

Migrating the NonStop AutoTMF Software Configuration to a New System

Three commands can assist you in migrating the NonStop AutoTMF software configuration to a new system:

- To migrate global parameters:

```
MODIFY GLOBALS REPLACE SYSTEMNUMBER n [, LISTALL ];
```

- To migrate Monitor and MapDB configuration:

```
MODIFY MAPDBS REPLACE NODENAME node [, LISTALL ];
```

- To migrate Program and File Attributes:

```
MODIFY AUTOTMF REPLACE NODENAME node [, LISTALL ];
```

The commands assume that SysDB tables, MapDB tables, configured object files and audited files are moved to same location on the new system. Only the system name and numbers are changed.

Migration Steps

The migration procedures vary depending on the placement of AutoTMF configuration files, object and audited files on the new system:

- The SysDB, MapDB, object files and audited files reside in the same location
- The SysDB and MapDB reside in a different location
- The object files or audited files reside in different locations

The SysDB, MapDB, object files and audited files reside in the same location

1. Install the NonStop AutoTMF software.
2. Create SysDB and license NonStop AutoTMF software.
3. Use the SQLCI COPY command to copy the content of the SysDB tables MAPDBS and REGISTRY into the new SysDB tables.
4. Use the SQLCI DUP command to move the MapDB tables ATMFATTR and ATMFPROG from original system to the new MapDB subvolume.
5. Issue MODIFY commands:
   - MODIFY MAPDBS
   - MODIFY GLOBALS
   - MODIFY AUTOTMF
6. Start the Monitor process.

The SysDB and MapDB reside in a different location

1. Install the NonStop AutoTMF software.
2. Create SysDB and license NonStop AutoTMF software.
3. Create MapDB with same options as the original system. This will automatically start the Monitor.
4. Stop Monitor.
5. In SQLCI, COPY SysDB tables MAPDBS and REGISTRY from original system into the newly creates SysDB tables.
6. In SQLCI, COPY MAPDB tables ATMFATTR and ATMFPROG from original system into the newly created MapDB tables.

7. Issue MODIFY commands:
   - MODIFY GLOBALS
   - MODIFY AUTOTMF

The object files or audited files reside in different locations

The user first proceeds as describe above.

The MODIFY AUTOTMF command makes the file and program attributes from source system visible to the INFO ATMFFILESET and INFO ATMFPROGRAMS commands. To complete the migration, proceed as follows:

1. Using Escort, capture the attributes of the object files and audited files configured on the original system using the INFO command specifying OBEYFORM and collect the output into an edit file using the LOG command:

```
LOG TO obeyfile;
INFO ATMFF,OBEYFORM;
INFO ATMFP,OBEYFORM;
LOG STOP;
EXIT;
```

2. Edit the log file *obeyfile* to change the location of the programs or audited files.
3. Using Escort, obey the modified *obeyfile* to configure the file attributes and program attributes on the new system:

```
OBEY obeyfile;
```

4. Optionally, delete the configured attributes for the original objects and audited files using the Escort DELETE ATMFPROGRAMS and DELETE ATMFFILESET commands.

Runtime Library

TNS Library

The NonStop AutoTMF software runtime library for TNS programs is an object file named ESCRUNTM; ESCRUNTM is provided as a user library and should be maintained in that form.

Acceleration

The runtime library is accelerated when it is shipped because it is performance sensitive. If the runtime library is, subsequently, combined with a user library (for example), the runtime library must be accelerated again.

On NonStop S-series servers, use AXCEL and specify the UL option as follows:

```
AXCEL $SYSTEM.ESCORT.ESCRUNTM, $SYSTEM.ESCORT.ESCRUNTM, UL;
```

Note that the accelerator may produce a few warnings.

On Integrity NonStop servers, use OCA as follows:

```
OCA $SYSTEM.ESCORT.ESCRUNTM, UL;
```

TNS/R Library

As discussed in Preparing TNS Programs on page 3-3, the NonStop AutoTMF software runtime library for TNS/R programs is distributed as four object files that correspond to the different classes of TNS/R programs:

- ESCRUNN (linkable TNS/R object) and ESCRUNNT (executable TNS/R object) for PTAL programs and for TNS/R C++ programs.
- ESCRUNNL (linkable TNS/R object) and ESCRUNNTM (executable TNS/R object) for TNS/R COBOL and C programs.

TNS/E DLL

The NonStop AutoTMF software runtime for native TNS/E programs is implemented as a DLL rather than a user library. The object file is ESCRUNDL.

Preparing a User Library

If some or all of your programs refer to a user library, the NonStop AutoTMF software runtime library should be bound with the existing user library object file. This step must be completed before you prepare the programs that reference the user library. The preparation of such programs will verify that the user library contains the NonStop AutoTMF software runtime. See Preparing Programs that Have a User Library on page 3-7 in Section 3, Preparing Programs for details.

PREPARE does not alter an existing reference to a user library, so you must replace each existing user library with one that contains the NonStop AutoTMF software runtime. You should not bind or link your user library routines to the standard $SYSTEM.ESCORT.ESCRUNTM object file because the standard runtime library is used by other NonStop AutoTMF software components.

Intercept Libraries

Intercept libraries should not be prepared. See paragraph Intercept Libraries in Section 3, Preparing Programs for details.

HIGHPIN Attribute

In order for a process to be run in a high pin (process number > 255), both the program object file and the user library file must have the HIGHPIN attribute set to ON. If either one is OFF, the process will occupy a low pin. The NonStop AutoTMF software user library files are distributed with HIGHPIN ON. If you bind ESCRUNTM or link ESCRUNN or ESCRUNNL with another user library, the resulting user library loses the HIGHPIN attribute.

- To set the HIGHPIN attribute for a TNS user library, use a Binder command, as follows:

```
BIND CHANGE HIGHPIN ON IN myuserlb
```

- To set the HIGHPIN attribute for TNS/R library file, use the NLD command as follows:

```
nld -change highpin on mynatlbt
nld -change highpin on mynatlbc
```

Hard-Coding Monitor Name into the Runtime

You can hard-code the monitor name into the NonStop AutoTMF runtime. This method adds the special procedure ESCORT_MONITOR_NAME into the AutoTMF runtime. Note that this method cannot be used for TNS/E native programs because the NonStop AutoTMF software product subvolume does not contain a linkable version of the AutoTMF DLL.

1. Create a source file with the following entries and assign the name of the alternate monitor to the *monname* variable ($ZTST in this example):

```
proc ESCORT_MONITOR_NAME( monname:len );
--   ------------------
string .ext monname;
int len;
begin
    monname ':=' "$ZTST";

end; -- ESCORT_MONITOR_NAME
```

2. Compile the source file using the appropriate compiler for each type of object file in your environment:

```
TAL  /IN mnames/mnameo;SYMBOLS,SUPPRESS
PTAL /IN mnames/mnamen;SYMBOLS,SUPPRESS,SRL
```

3. BIND and LINK the procedure into the runtime object files

```
In Binder:
select LIST * OFF
SELECT CHECK PARAMETER STRONG
SELECT SATISFY OFF
ADD * FROM $DATA.ESCORT.ESCRUNTM
ADD * FROM mnameo, DELETE
SET LIKE $DATA.ESCORT.ESCRUNTM
SET HIGHPIN ON
BUILD ESCRUNTM !
```

```
nld mnamen $data.escort.escrunnl &
-o $data.escort.escrunnm -ul -no_data_rearrange &
-allow_duplicate_procs

nld mnamen $data.escort.escrunn -o $data.escort.escrunnt -ul &
-no_data_rearrange -allow_duplicate_procs
```

Updating the Runtime Library

Whenever you need to replace or alter the NonStop AutoTMF software runtime library object files, you should make sure that all processes using ESCRUNTM, ESCRUNNT or ESCRUNNM as a user library have been stopped. The CI command INFO LIBRARY has been provided to display and, optionally, to stop processes using a specified user library.

If you update ESCRUNTM, ESCRUNNT or ESCRUNNM without stopping processes, through file renaming, errors may occur. If a program is executing with the user library that has been renamed and another process is started with the same program, the operation will often fail.

Host-language Runtimes in the Runtime Library

TNS Host Runtime Language Libraries

To support COBOL and C programs, as well as TAL programs using the SIO and CRE libraries, copies of the standard host-language runtimes are bound into the NonStop AutoTMF software runtime library ESCRUNTM. These libraries are exact copies of the libraries distributed on a site update tape, except that selected procedure calls are retargeted to NonStop AutoTMF software intercept procedures.

Table A-1. HP Language Runtime Libraries included in the NonStop AutoTMF Software Runtime

Product	Product Number	Release Subvol	File
COBOL 85	T9267	ZCOBOL85	C8LIB
CRE	T9280	ZCRERTL	CFELIB CREKERN
C	T9549	ZC	CLIB
GPLIB	T9600	ZGPLIB	GPLIBXR GPLIBR (INITIALIZER only)

COBOL 74 (T9251/T9261) is not currently supported.

Host Language Runtime Library Versions

The NonStop AutoTMF software install process will select the proper version of the NonStop AutoTMF software runtime library that is compatible with your Nonstop OS version. Each NonStop AutoTMF software runtime contains the latest planned product maintenance or unrestricted time-critical fix software product revisions (SPRs) for each host-language runtime.

If, however, you have a dependency on a particular SPR level for any of the above products and need to install an SPR for TNS or TNS/R native COBOL, C, CRE, or GPLIB in the NonStop AutoTMF software runtime, please contact product support to obtain an updated version of the runtime object files.

The softdoc file describes the SPR levels for each host-language runtime that is included in the runtime library. To verify which release of the host-language library files is present in the runtime, use the VPROC utility:

```
69> vproc $system.escort.escruntm
VPROC - T9617G03 - (07 AUG 2003) SYSTEM \NSKNED   Date 20 JUL 2005,
16:35:47
Copyright 2003 Hewlett-Packard Development Company, L.P.


$SYSTEM.ESCORT.ESCRUNTM
     Binder timestamp:  27JUN2005 14:31:58
    Version procedure:  S7053D45^14FEB01^LOAD^AAX^04JUN
    Version procedure:  T0581H01^01MAY2005^ATM170^H01
    Version procedure:  T0581H01^01MAY2005^H01
    Version procedure:  T0581H01^01MAY2005^RUN170^H01
    Version procedure:  T0581V03^01MAY2005^H01
    Version procedure:  T9267D46^19AUG2002^05AUG02^ADU
    Version procedure:  T9280D40^08JUL2002^AAX^KERN
    Version procedure:  T9280D40^08JUL2002^CFE
    Version procedure:  T9549D30^08APR2000^AAK
    Version procedure:  T9600D40^22JAN03^GPLIBX^07JAN03
           Target CPU:  UNSPECIFIED
      AXCEL timestamp:  28JUN2005 14:16:58
```

TNS/R Host Runtime Language Libraries

As described in Preparing TNS Programs, there are two classes of TNS/R NonStop AutoTMF software runtime libraries: ESCRUNN/ESCRUNNT for PTAL only programs, ESCRUNNL and ESCRUNNM for COBOL and C programs.

The following tables describe the host-language product release files that are included in the NonStop AutoTMF software TNS/R runtime libraries:

Table A-2. TNS/R HP Products in ESCRUNN and ESCRUNNT

Product	Product Number	Release Subvol	File
TNS/R GPLIB	T9600	ZGPLIB	GPLIBX GPLIB (INITIALIZER only)

Table A-3. TNS/R HP Products in ESCRUNNL and ESCRUNNM

Product	Product Number	Release Subvol	File
TNS/R COBOL	T8108	ZRELSRL	ZCOBREL
TNS/R CRE	T8431	ZRELSRL	ZCREREL
TNS/R C	T8432	ZRELSRL	ZCRTLREL
TNS/R GPLIB	T9600	ZGPLIB	GPLIBX GPLIB (INITIALIZER only)

TNS/R Host Language Runtime Library Versions

The TNS/R NonStop AutoTMF software runtime contains the latest planned product maintenance or unrestricted time-critical fix software product revisions (SPRs) for each host-language runtime D46/G06 and later RVUs of the NonStop OS.

The SPR levels for each host-language runtime that is included in the runtime library are documented in the softdoc file. To verify which version of the host-language library files are present in the runtime, use the VPROC utility:

```
47>  vproc escrunnt
VPROC - T9617G03 - (07 AUG 2003) SYSTEM \NSKNED   Date 20 JUL 2005,
16:37:05
Copyright 2003 Hewlett-Packard Development Company, L.P.

$SYSTEM.ESCORT.ESCRUNNT
 GMT Binder timestamp:  27JUN2005 21:32:16
    Version procedure:  T0581H01^01MAY2005^RUN170^H01
    Version procedure:  T9600D40^22JAN03^GPLIB^07JAN03
    Version procedure:  T9600D40^22JAN03^GPLIBX^07JAN03
          Native Mode:  runnable file
```

```
48> vproc escrunnm
VPROC - T9617G03 - (07 AUG 2003) SYSTEM \NSKNED   Date 20 JUL 2005,
16:38:39
Copyright 2003 Hewlett-Packard Development Company, L.P.

 GMT Binder timestamp:  27JUN2005 21:32:11
    Version procedure:  T0581H01^01MAY2005^RUN170^H01
    Version procedure:  T9600D40^22JAN03^GPLIB^07JAN03
    Version procedure:  T9600D40^22JAN03^GPLIBX^07JAN03
    Version procedure:  T8431G08^01AUG2001^AAF
    Version procedure:  T8108D46^19AUG2002^30JUL02^ABB
    Version procedure:  S7053D45^14FEB01^LOAD^AAX^04JUN
    Version procedure:  T8432G08_01FEB2001_ZCRTLSRL
          Native Mode:  runnable file
```

TNS/E Host Language Runtime DLLs

The COBOL and C runtimes are implemented as the public DLLs ZCOBDLL and ZCRTLDLL. Unlike the TNS and TNS/R versions of the NonStop AutoTMF software runtime, the TNS/E runtime does not contain copies of the host language runtime. Instead, to perform the procedure call intercepts, prepared copies of the currently installed public host language DLLs are moved and “rebased” to the AutoTMF operational subvolume at installation time. The rebased public DLLs are then used as private DLLs that are set up to use the NonStop AutoTMF runtime as a DLL.

The public DLLs have dependencies on other public DLLs. To properly intercept the procedure calls the public DLLs issue, the INSTALL macro currently copies and rebases the following public DLLs:

Table A-4. Public DLLs rebased and prepared for NonStop AutoTMF

Public DLL	Referenced by
ZCOBDLL	rZCREDLL
ZCREDLL	
ZCRTLDLL	ZCREDLL,ZI18NDLL,ZICNVDLL,ZOSSKDL
ZI18NDLL	ZCREDLL,ZCRTLDLL,ZICNVDLL
ZICNVDLL	ZCREDLL,ZCRTLDLL,ZI18NDLL
ZOSSKDLL	ZCREDLL,ZCRTLDLL

Other public DLLs may be rebased in future versions of NonStop AutoTMF.

New Versions of DLLs and Operating System Upgrades

If you install a new version of an existing rebased public DLL (COBOL for example), or a new version of the Operating System, you must update the DLL (or DLLs) in the AutoTMF operational subvolume.

You can perform the DLL rebasing and preparing steps manually, as described in paragraph DLL Update Steps, or use the UPDDLLS macro to update all the DLLS required for AutoTMF, as described in paragraph Running UPDDLLS.

DLL Update Steps

1. Locate the DLL to install. The file ZREGPTR in $SYSTEM.SYS*nn* contains the name of the subvolume where the currently installed DLLs are running.
2. Use INFO PROGRAM (or PROGINFO (PI)) with the REBASE option to generate the ELD command that copies and rebases a public DLL to the AutoTMF operational subvolume.
3. Run the generated command.
4. Prepare the rebased DLL.

The example below installs a new version of the COBOL DLL:

```
AutoTMF 1?proginfo $SYSTEM.ZDLL044.ZCOBDLL,rebase;
ELD -alf $SYSTEM.ZDLL044.ZCOBDLL -t 0 -d D0000 -o $SYSTEM.ESCORT.zcobdll

AutoTMF 2? ELD -alf $SYSTEM.ZDLL044.ZCOBDLL -t 0 -d D0000 -o
$SYSTEM.ESCORT.zco
bdll

eld - TNS/E Native Mode Linker - T0608H01 - 05APR05
Copyright 2005 Hewlett-Packard Company
This program may be distributed under the terms of the GNU
General Public License.

eld command line:
   \drpsoc.$system.system.eld -alf $SYSTEM.ZDLL044.ZCOBDLL -t 0 -d D0000 -
o
   $SYSTEM.ESCORT.zcobdll

**** INFORMATIONAL MESSAGE **** [1019]:
   Using DLL $SYSTEM.zdll044.zcredll.
**** INFORMATIONAL MESSAGE **** [1530]:
   Using the zimpimp file $SYSTEM.SYS00.ZIMPIMP.

Output file: $SYSTEM.ESCORT.zcobdll (dll)
Output file update timestamp: May  4 13:40:25 2005

No errors reported.
No warnings reported.
2 informational messages reported.
Elapsed Time:  00:00:02
AutoTMF 3? prepare zcobdll, ul;
--- $SYSTEM.ESCORT.ZCOBDLL preparation complete
--- AutoTMF runtime library $SYSTEM.ESCORT.ESCRUNDL
```

Running UPDDLLS

The TACL macro UPDDLLS performs the rebasing and preparing steps described above, for all the required DLLs, much like the INSTALL macro does when a new version of NonStop AutoTMF is installed.

To rebase and prepare all the host language DLLs listed in Table A-4:

1. Volume to the operational product subvolume ($system.escort for example)
2. RUN UPDDLLS

The macro displays the timestamp of the DLLs in the product subvolume and the timestamp of the system DLLs. It then prompts you to confirm you wish to proceed with the updates.

The example below shows the execution of UPDDLLS in $SYSTEM.ESCORT:

```
13> volume $system.escort
14> run upddlls
HP Nonstop(tm) AutoTMF(tm) DLL Update Installer

Rebase and Prepare Public DLLs...

    DLL         Current Binder Timestamp     New Binder Timestamp
    ---         ------------------------     --------------------

  ZOSSKDLL        28JUN2010  12:16:53         11FEB2011  17:21:55
  ZICNVDLL        06MAR2010  15:07:11         06MAR2010  15:07:11
  ZI18NDLL        08MAR2010  09:33:52         08MAR2010  09:33:52
  ZCRTLDLL        28JUN2010  19:51:19         02SEP2010  19:23:59
  ZCREDLL         29JUN2010  22:30:21         29JUN2010  22:30:21
  ZCOBDLL         11MAY2010  21:32:46         23DEC2010  01:18:07

 Enter Y to update these DLLs or any other key to exit: y

$$system.ZDLL009A.ZOSSKDLL rebased to $SYSTEM.ESCORT.ZOSSKDLL and prepared
$system.ZDLL009A.ZICNVDLL rebased to $SYSTEM.ESCORT.ZICNVDLL and prepared
$system.ZDLL009A.ZI18NDLL rebased to $SYSTEM.ESCORT.ZI18NDLL and prepared
$system.ZDLL009A.ZCRTLDLL rebased to $SYSTEM.ESCORT.ZCRTLDLL and prepared
$system.ZDLL009A.ZCREDLL rebased to $SYSTEM.ESCORT.ZCREDLL  and prepared
$system.ZDLL009A.ZCOBDLL rebased to $SYSTEM.ESCORT.ZCOBDLL  and prepared

The Public DLLs have been rebased for use by AutoTMF.
```

Transporting Applications

If you have a multi-system environment and move NonStop AutoTMF software-prepared applications from one system to another, you do not need to prepare the programs again. However you must ensure that the NonStop AutoTMF software runtime user library is properly configured. Therefore, you should install NonStop AutoTMF software in the same subvolume on every system; the default $SYSTEM.ESCORT is a good choice.

If you do need to alter the user library for a collection of programs, consider using the NSKFIXUP command.

Version Checking

Each of the primary software components contains version checking routines to ensure compatibility between the components. In particular, the software makes the following version checks:

1. When the CI is started, it obtains the version of SysDB and MapDB. If the versions do not match, you might be able to upgrade the affected tables. Consult the NonStop AutoTMF software softdoc file if this occurs.

2. The CI also requests the version from the selected monitor process, which must match the expected version in the CI.
3. Whenever the CI opens a new monitor, the version checks are made for both the monitor process and the associated MapDB.
4. When a START MONITOR command is run, the monitor terminates if the CI version does not match the monitor's version. The monitor also checks the version of MapDB.
5. When an application process is run, each request from the NonStop AutoTMF software runtime to the monitor contains a version number. If the version does not match the monitor's version, the request is rejected, and the application terminates.

Security and Availability

This section describes various security and availability considerations when installing and using Escort. This material should be studied carefully before converting a production application to use Escort.

Note. Requirements for security vs. ease-of-access vary from customer to customer. The following security guidelines may or may not apply to your security requirements, You are ultimately responsible for ensuring the appropriate level of security.

Overview

Security considerations are essentially two-fold:

- When used in production, NonStop AutoTMF software becomes an essential part of the application environment; damage or misuse of the product may result in an application outage and business disruption. You should take steps to secure the product to ensure its continuous availability.
- Since production database accesses are intercepted by NonStop AutoTMF software, there is an opportunity for misuse that could compromise security of your database. You can prevent security breaches with a few simple installation and configuration steps.

In general, a few minor security considerations are introduced, but the product does not represent a major avenue for breaches of system or data security. NonStop AutoTMF software is not privileged and does not require use of the SUPER.SUPER account at any time. It depends on conventional file system security to protect its own configuration data. With the exception of the tracing facility, described below, it depends on conventional file system security to protect your database and other files.

Data File Access Security

All application data access is performed using standard, non-privileged, operations from the application process. The form of access may be altered (such as to perform

additional Enscribe operations), but these operations are subject to the same security control as provided for all application access to data.

Object File Access Security

Normally, only *execute* access authority is required to run a program. With NonStop AutoTMF software, however, both *read* and *execute* access authority is required to run a prepared program.

The NonStop AutoTMF software runtime needs to determine if a program has embedded SQL and a few other important facts obtained by reading the object file. Thus, you must give each user who runs an object file both *execute* and *read* access authority to the file; *write* access authority is not required.

To prepare an application program to use NonStop AutoTMF software, the object file is modified to intercept procedure calls and assign a user library; this requires *write* access authority.

Configuration Security

When used in production, NonStop AutoTMF software becomes an essential part of the application and must be protected against damage or misuse. Ideally, the installation should have a designated person or small group that is responsible for performing basic maintenance of the NonStop AutoTMF software environment.

Product Security

The files in the product subvolume should be managed in a manner similar to the system components in $SYSTEM.SYSTEM. The files should be protected against change or deletion by unauthorized persons.

System Database Security

The System Database (SysDB) contains the licensing information and many global settings that will affect all applications that are prepared to use Escort. To prevent inadvertent changes to the SysDB, you should limit write access authority to the users that are responsible for maintaining the NonStop AutoTMF software environment. This is usually accomplished when the SysDB is created (see the CREATE SYSDB on page 6-54) or by use of a SQLCI ALTER TABLE command for both the REGISTRY and MAPDBS tables.

Application programs have no need to access the SysDB; the monitor process provides all the SysDB information required by applications.

Mapping Database Security

Security of the Mapping Database (MapDB) is less of an issue than the security of the SysDB as long as the personnel accessing and updating it can be expected to act

responsibly. Often, personnel in operations, development, testing, and production control update the MapDB for their own file and program entities.

On the other hand, damage or destruction of the MapDB will also result in an outage of the application, so you might want to restrict update access of the MapDB to trusted users.

Application programs have no need to access the MapDB; the monitor process provides all the MapDB information required by applications.

Monitor Process Security

The monitor process is an essential part of the application environment. All applications obtain configuration information from the monitor. If the monitor is not available, applications cannot run properly and an application outage occurs.

The monitor process is a fault-tolerant process pair. The most likely cause of unavailability would be the accidental stopping of the process by either the TACL STOP or the CI STOP MONITOR commands.

To guard against accidental stopping, the process should be started by a trusted user.

All CI commands that might affect the availability of the monitor process, including STOP MONITOR, have been restricted to the user that started the monitor process or a member of the SUPER group.

Configuration Backups

Since the configuration is critical to application availability, it is recommended that the user perform a BACKUP of the NonStop AutoTMF, SysDB, and MapDB subvolumes on a regular basis.

Tracing and Debugging Security

The tracing facility is a powerful tool than can cause programs to produce a trace of their Enscribe and TMF operations (including data records accessed) and to enter Debug/Inspect when the process is started. Users will find this facility useful for diagnosing suspected application errors, both those that may be due to NonStop AutoTMF software as well as ordinary application program errors.

Tracing and debugging can be a significant source of data security exposure, allowing unauthorized persons to view sensitive data. Such exposure can be prevented using NonStop AutoTMF software configuration parameters.

The tracing facility can be invoked in two ways:

- A user issues a TRACE command that causes a selected process or processes to be traced or enter debug. Some other user starting a process may not know that the process is being traced or debugged.

- A user specifies DEFINEs, such as =_ESCORT_STATEMENT_TRACE when executing the process; in this case, the user has control over tracing and debugging.

The TRACE command can be secured to prevent exposure, as follows:

- The global setting for SECURETRACE should be ON; use the ALTER GLOBAL command to set the SECURETRACE setting. The SysDB should be suitably secured to prevent unauthorized users from altering this setting.

  When SECURETRACE is ON, the user who issues the TRACE command must be the process access ID of the processes that are to perform tracing. Other processes will run normally and will not trace or enter debug.

  In development and testing systems, or systems where security is not a major issue, you may wish to set SECURETRACE to OFF to facilitate problem analysis.

- When a user wishes to trace DATA (that is, produce a trace in which the images of data records accessed by the program are displayed), NonStop AutoTMF software requires the trace file to be secured so that only the process access id of the process may read the file. This ensures that no unauthorized person can read the database records in the trace.

Executing Prepared Applications

Swap space

When an application process first opens a file, the NonStop AutoTMF software runtime allocates an extended memory segment, which requires a disk swap file. Space allocation or volume security errors may cause the allocation to fail, which will terminate the application process.

The default location for a swap file is the volume on which the object file resides. If the space on that volume is insufficient, the user can allocate up to four alternate swap volumes using the global parameters SWAPVOL[0],SWAPVOL1, SWAPVOL2, SWAPVOL3.

You can also set the global parameter KMSF to enable Kernel Managed Swap Facility (KMSF) allocation in the

See ALTER GLOBAL on page 6-34 for details about SWAPVOL*n* and KMSF parameters.

EMS Logging

In addition to being displayed on the home terminal of the application program, the internal errors detected by the runtime are logged to an EMS collector. The default collector is $0. Logging to an alternate collector can be configured using the global parameter EMSCOLLECTOR or, if required on a process by process basis, by using a

DEFINE. See ALTER GLOBAL on page 6-34 and =_ESCORT_EMS_COLLECTOR on page B-6 for details.

A template called ZESCTMPL is shipped with the product and can be combined with existing templates to use with Viewpoint or any EMS distributor.

The EMS filter file ESCFLTR is supplied in the product subvolume to configure an EMS distributor process to display NonStop AutoTMF software events.

B Special DEFINEs

Introduction

You can use Guardian DEFINEs to enable a number of NonStop AutoTMF software optional features or to provide diagnostic and management information. Defines are useful to test a NonStop AutoTMF software feature for a selected program, without altering the existing configuration.

Configure NonStop AutoTMF software features using the CI commands provided when possible, because this is easier and less error-prone than using DEFINEs. DEFINEs must be inserted in each PATHWAY configuration and batch program execution setup; they are very easy to omit and diagnosing a missing DEFINE is difficult.

You can configure most of these features by setting the global parameters, or for selected files by setting a parameter of AUTOTMFFILESET, or for selected programs by setting a parameter of AUTOTMFPROGRAMS. See ALTER GLOBAL on page 6-34, ADD ATMFFILESET on page 6-6, and ADD ATMFPROGRAMS on page 6-12 for details. If specified, a DEFINE overrides configured settings.

This appendix catalogs all DEFINEs that have some direct effect on NonStop AutoTMF software. Other DEFINEs may also be used for normal application file assignments.

DEFINE Types

In many cases, the mere presence of a DEFINE will select the optional feature. Such a DEFINE should be of class MAP and specify any legal file name; the file does not need to exist.

In other cases, the file name of a DEFINE is an encoded message; again, the file does not need to exist.

A third type of DEFINEs specifies actual file names for tracing and the like.

The last type of DEFINE specifies a subvolume; these should be of class CATALOG.

Table B-1. Runtime DEFINES

DEFINE Name	Description
=_ESCORT_ATMF_ISOLATION	Sets transaction isolation level.
=_ESCORT_ATMF_MAXTIME	Sets the maximum duration of automatic transactions.
=_ESCORT_ATMF_MAXUPDATE	Sets the maximum number of updates per automatic transaction.
=_ESCORT_ATMF_NOWAIT	Instructs NonStop AutoTMF software to commit automatic transaction in a nowaited mode.

Table B-1. Runtime DEFINES (continued)

DEFINE Name	Description
= ESCORT_ATMF_OFF	Disables NonStop AutoTMF software for a prepared program.
= ESCORT_ATMF_TXHOLDOFF	Specifies inter-request delay for holding transactions in active state.
= ESCORT_ATMF_WAITED	Reverses the nowait mode set for completion of automatic transactions.
= ESCORT_AUDIT_RENAME	Enables the renaming of audited files.
= ESCORT_DYNAMIC_TRC_ON	Enables dynamic tracing of a prepared program.
= ESCORT_EMS_COLLECTOR	Sends runtime messages to the EMS collector.
= ESCORT_MONITOR	Sets the Monitor-MapDB pair for the execution environment.
= ESCORT_OPTIMIZEUNLOCKS	Enables UNLOCKFILE optimization.
= ESCORT_OPTMZUNLOCKSOFF	Disables UNLOCKFILE optimization.
= ESCORT_READ_NULL_RECS	Reverses the effect of = ESCORT_SKIP_NULL_RECS
= ESCORT_SKIP_NULL_RECS	Skips zero-length records in audited entry-sequenced files
= ESCORT_STATEMENT_DATA	Traces the Enscribe calls issued by the application and the data read and written to the data files.
= ESCORT_STATEMENT_KEYS	Traces the Enscribe calls issued by the application and the key data read and written to the data files.
= ESCORT_STATEMENT_TRACE	Displays execution of Enscribe calls issued against Enscribe files.
= ESCORT_SUPPRESS_AUDIT	Ignores transactions generated by a user process.
= ESCORT_SUPPRESS_INHRTX	Ignores transactions inherited on $RECEIVE.

Table B-2. Command Interpreter DEFINEs

DEFINE Name	Description
= ESCORT_MONITOR	Sets the monitor-MapDB pair for the execution environment.
= ESCORT_SYSDB	Specifies an alternate SysDB. (For testing only)

Runtime DEFINES

=_ESCORT_ATMF_ISOLATION

You can set the transaction isolation level for a program by specifying the =_ESCORT_ATMF_ISOLATION DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_ISOLATION, CLASS MAP, FILE value

where value is one of WEAK, NORMAL or STRONG
```

=_ESCORT_ATMF_MAXTIME

You can set the maximum duration for automatic transactions with the =_ESCORT_ATMF_MAXTIME DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_MAXTIME, CLASS MAP, FILE value

value is Snnn where n is 1 to 4096
```

The numeric portion of *value* is the maximum durations of an automatic transaction in seconds. The value of *n* should be a power of 2 (1,2,4,8,16,32,...). If *n* is not entered as such, the value will be rounded to a power of 2.

This DEFINE does not override the AUTOTMFFILESET parameter if a file is configured for SEPARATETX.

=_ESCORT_ATMF_MAXUPDATE

You can set the maximum number of updates per automatic transaction with the =_ESCORT_ATMF_MAXUPDATE DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_MAXUPDATE, CLASS MAP, FILE value

value is Snnn where n is 1 to 4096
```

The numeric portion of *value* is the maximum number of updates in a single transaction. The value of *n* should be a power of 2 (1,2,4,8,16,32,...). If *n* is not entered as such, the value will be rounded to a power of 2.

This DEFINE does not override the AUTOTMFFILESET parameter if a file is configured for SEPARATETX.

=_ESCORT_ATMF_NOWAIT

you can commit automatic transactions in a NOWAIT fashion with the =_ESCORT_ATMF_NOWAIT DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_NOWAIT, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored.

This DEFINE does not override the AUTOTMFFILESET parameter if a file is configured for SEPARATETX.

=_ESCORT_ATMF_OFF

You can disable NonStop AutoTMF software for a process with the =_ESCORT_ATMF_OFF DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_OFF, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored.

=_ESCORT_ATMF_TXHOLDOFF

By default, automatic transactions are committed when a server process replies to a request. You can cause a transaction to be kept active between requests with the =_ESCORT_ATMF_TXHOLDOFF DEFINE. The transaction will usually be committed based on the MAXTIME and MAXUPDATES parameters, or if the delay between requests exceeds a value set by the DEFINE.

The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_TXHOLDOFF, CLASS MAP, FILE value

value is Sn where n is 0 to 120 (seconds)
```

The numeric portion of the file name is the maximum inter-request delay.

This NonStop AutoTMF software feature was designed for certain types of batch processing that is implemented as a master process that sends requests to server processes. If the batch process is restarted after a failure, it may not matter if some number of completed server requests are aborted.

When the master process stops sending work to the server, there may be an active transaction, record locks, and uncommitted updates. A commit timer will commit the active transactions after a specified idle period. To be effective, the master process may send a terminating request to the servers. The master process must not stop the server processes.

Automatic transactions are committed if:

- the process terminates (but not if the process is stopped), or
- the server is waiting for a request and the time since the last request exceeds the specified value, or
- the transaction has reached the limits set by the MAXTIME and MAXUPDATES parameters.

Each transaction will span many server requests and updated records will be kept locked between server requests. If the batch processing sends work to many server processes, they must access different database records or a deadlock may occur. The commit timer will not cause the current transaction to be committed if the process is deadlocked.

=_ESCORT_ATMF_WAITED

You can reverse a configuration for NOWAIT transactions with the =_ESCORT_ATMF_WAITED DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_ATMF_WAITED, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored. This DEFINE does not override the AUTOTMFFILESET parameter if a file is configured for SEPARATETX.

=_ESCORT_AUDIT_RENAME

You can enable renaming of audited files with the =_ESCORT_AUDIT_RENAME DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT__AUDIT_RENAME, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored.

=_ESCORT_DYNAMIC_TRC_OFF

You can disable dynamic tracing of a program with the =_ESCORT_DYNAMIC_TRC_OFF DEFINE. The process will not check the state of the trace signaling file to determine if tracing has been requested; all trace requests must be active before the process starts. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_DYNAMIC_TRC_OFF, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored.

=_ESCORT_DYNAMIC_TRC_ON

You can enable dynamic tracing of a program with the =_ESCORT_DYNAMIC_TRC_ON DEFINE. The program periodically checks the state of a signaling file to determine if tracing has been requested and will start tracing at that point. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_DYNAMIC_TRC_ON, CLASS MAP, FILE $X.Y.Z
```

The FILE parameter is ignored.

=_ESCORT_EMS_COLLECTOR

You can select an alternate destination for EMS event messages with the =_ESCORT_EMS_COLLECTOR DEFINE. These events generally pertain to configuration problems. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_EMS_COLLECTOR, CLASS MAP, FILE $collector
```

The process name specified with the FILE parameter must be an EMS collector, having a device type 1 and subtype 0, otherwise the DEFINE is ignored.

Messages are also sent to the process home terminal.

=_ESCORT_MONITOR

See description of =_ESCORT_MONITOR on page B-10 below.

=_ESCORT_OPTIMIZEUNLOCKS

Specify this define to enable the optimization of UNLOCKFILE operations, which is equivalent to configuring the ATMFPROGRAMS attribute OPTIMIZEUNLOCKS.

```
ADD DEFINE =_ESCORT_OPTIMIZEUNLOCKS, CLASS MAP, FILE $X.Y.Z
```

The file parameter is ignored.

Non TMF-aware applications sometimes issue a blanket call to UNLOCKFILE to release all locks rather than managing locks individually. When optimization is enabled:

- NonStop AutoTMF software eliminates the call to UNLOCKFILE.
- Since the program has released all locks on the file, NonStop AutoTMF software attempts to commit the transaction, subject to the usual protocol for committing automatic transactions (no other lock is held on another file participating in the transaction).

Note that the optimization can only be enabled for files that are also enabled for automatic transactions.

=_ESCORT_OPTMZUNLOCKSOFF

Specify this define to disable the optimization of UNLOCKFILE operations.

```
ADD DEFINE =_ESCORT_OPTMZUNLOCKSOFF, CLASS MAP, FILE $X.Y.Z
```

The file parameter is ignored.

This define overrides the value of the global parameter ATMFOPTIMIZEUNLOCK or of the ATMFPROGRAMS attribute OPTIMIZEUNLOCKS.

=_ESCORT_READ_NULL_RECS

You can request that the NonStop AutoTMF software runtime return (not skip) zero-length records when a process is reading sequentially through an audited entry-sequenced. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_READ_NULL_RECS, CLASS MAP, FILE $X.Y.Z
```

The file parameter is ignored.

This define reverses the effect of the =_ESCORT_SKIP_NULL_RECS define. See the description below for details.

Specifying this define overrides the value of the global parameter ATMFSKIPNULLRECS.

=_ESCORT_SKIP_NULL_RECS

You can request that the NonStop AutoTMF software runtime ignore zero-length records when a process is reading sequentially through an audited entry-sequenced file. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_SKIP_NULL_RECS, CLASS MAP, FILE $X.Y.Z
```

The file parameter is ignored.

If a record is inserted into an audited entry-sequenced file and subsequently backed out because a transaction is aborted, a zero length record is left in the file where the record had been inserted. This does occur if the file is not audited and can cause problems for programs that are not expecting to encounter such records.

Specifying this define precludes the program from reading unexpected null records.

Although automatic transactions are never aborted by NonStop AutoTMF software, unilateral aborts could leave such gaps in an entry-sequenced file.

Specifying this define overrides the value of the global parameter ATMFSKIPNULLRECS.

=_ESCORT_STATEMENT_DATA

You can request a process trace that includes data records with the =_ESCORT_STATEMENT_DATA DEFINE. The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_STATEMENT_DATA, CLASS MAP, FILE filename
```

This define traces a superset of the information traced by the =_ESCORT_STATEMENT_TRACE. See the description of that DEFINE below.

=_ESCORT_STATEMENT_KEYS

You can request a process trace that includes record key fields with the =_ESCORT_STATEMENT_KEYS DEFINE.

The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_STATEMENT_KEYS, CLASS MAP, FILE filename
```

This define traces a superset of the information traced by the =_ESCORT_STATEMENT_TRACE. See the description of that DEFINE below.

=_ESCORT_STATEMENT_TRACE

You can request a process trace with the =_ESCORT_STATEMENT_KEYS DEFINE.

Each operating system procedure, such as OPEN, KEYPOSITION, READ and WRITE, that is intercepted by the NonStop AutoTMF software runtime library is traced. The trace is detailed and can include key fields and entire records if so configured. In addition, the trace contains much of the process startup information, such as

- The startup message
- ASSIGNs passed to the process
- DEFINEs active when the process started

Each entry is timestamped and contains the name (or CPU, pin) of the calling process.

The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_STATEMENT_TRACE, CLASS MAP, FILE filename
```

The file specified by this DEFINE is opened when the process is executed. The file should be a process, a terminal, an entry-sequenced file with a record length of at least 132 bytes, or an Edit file.

If a disk file does not exist, the file is created. The trace file is opened shared and existing data is not purged. A trace file can be used by many processes concurrently.

Edit trace files provide for the most efficient tracing, but an Edit trace file can only be opened by one process at a time.

Tracing can be configured easily, and with more options, with the NonStop AutoTMF software command interpreter; see TRACE on page 6-81. Using the TRACE command does not require changing the setup for executing the processes to be traced.

=_ESCORT_SUPPRESS_AUDIT

This define causes NonStop AutoTMF software to ignore transactions generated by a user process, which is equivalent to configuring the ATMFPROGRAM attribute SUPPRESSUSERTX.

```
ADD DEFINE =_ESCORT_SUPPRESS_AUDIT, CLASS MAP, FILE $X.Y.Z
```

The file specified is ignored.

This define is supplied for testing purposes only. Programs that make sophisticated use of TMF depend on the correct operation of the calls that are eliminated by this option. Enabling this option may cause program failures and data corruption.

When specified, this define causes the NonStop AutoTMF software runtime to behave as follows:

- All BEGINTRANSACTION and ENDTRANSACTION operations requested by the application are ignored.
- All transactions inherited from requesters through messages on $RECEIVE are ignored.
- FILEINFO requests on audited files report that files are not audited.

=_ESCORT_SUPPRESS_INHRTX

This define causes NonStop AutoTMF software to ignore transactions inherited by a process through $RECEIVE, which is equivalent to configuring the ATMFPROGRAMS attribute SUPPRESSINHERITEDTX.

```
ADD DEFINE =_ESCORT_SUPPRESS_INHRTX, CLASS MAP, FILE $X.Y.Z
```

The file specified is ignored.

A non TMF-aware server may inherit a transaction from a TMF-aware requester when reading $RECEIVE. If NonStop AutoTMF software detects an inherited transaction, NonStop AutoTMF software assumes that the process uses that transaction to access audited files. Suppressing inherited transactions insures that NonStop AutoTMF software generates automatic transactions for all audited file accesses, emulating the behavior of a non TMF-aware process.

Command Interpreter DEFINEs

=_ESCORT_SYSDB

This DEFINE is used for testing NonStop AutoTMF software. It permits the user to establish an independent NonStop AutoTMF environment on the same system. In general, its use is not required and is not recommended in a production environment. In order to test a new version on NonStop AutoTMF software in parallel with an existing version, you should use an alternate monitor and MapDB instead.

The =_ESCORT_SYSDB DEFINE is class CATALOG and specifies a subvolume for the SysDB tables. If this DEFINE is present when the CI is started, the CI will not look for the normal SysDB, but will use the one specified. Thus, the CI will operate only with MapDBs and monitor processes that were created under the specified SysDB.

=_ESCORT_SYSDB DEFINE has no effect on an application, since the DEFINE obtains all configuration information from a monitor process. To run an application in the separate environment, the =_ESCORT_MONITOR DEFINE selects a monitor process defined in that environment.

A typical =_ESCORT_SYSDB DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_SYSDB, CLASS CATALOG, SUBVOL $vol.mysysdb
```

=_ESCORT_MONITOR

This DEFINE selects a monitor process for the CI session; using this DEFINE is equivalent to specifying the monitor name on the command line or issuing an OPEN on the monitor name.

The DEFINE is added with the following TACL command:

```
ADD DEFINE =_ESCORT_MONITOR, CLASS MAP, FILE $xxxx
```

Alternate MapDBs are typically created to test a new version of NonStop AutoTMF software. By issuing the INFO MAPDB * or STATUS MONITOR * commands in the CI, you can determine the currently defined monitor process names. A monitor process name must contain exactly 5 characters, including the $. The default monitor process name is $ZESC.

C Problem Resolution

HP NonStop AutoTMF software is designed to be invisible to an application program. NonStop AutoTMF software is logically a layer that looks like the file system to the programs and like an application to the file system. Because the runtime is a library to the application programs, its operating environment is the application's environment; thus, distinguishing between application problems and NonStop AutoTMF software problems may be difficult.

This appendix discusses how to diagnose problems and the tools to do so and how to report them given the context of NonStop AutoTMF software applications. This appendix covers the following topics:

Runtime Errors

Runtime errors fall into the following categories:

Program Failures (ABEND)

NonStop AutoTMF software intercepts Enscribe I/O and selected other operating system calls. It tracks the sequence of calls and makes decisions about generating and committing automatic transactions. It then issues the Enscribe calls on behalf of the application program.

Errors returned from the procedure calls are passed back unchanged to the application program. The application then proceeds with its normal error checking and recovery.

TMF errors

The introduction of TMF transactions may cause programs to encounter errors that they are not expecting and that they are unable to handle. In most cases, such conditions, when detected by NonStop AutoTMF software, cause a program to abend immediately to avoid compromising data integrity.

You can configure NonStop AutoTMF software to stop a program rather than causing it to abend when a program encounters an TMF environmental error, by setting the global parameter ATMFSTOPONTMFERR or the program attribute STOPONTMFERR to ON. AutoTMF stops the program and no saveabend file is produced.

If NonStop AutoTMF software stops or aborts a program due to a TMF environmental error, a message describing the error sent to the EMS log to alert the operator of the failure.

Other errors

Program failures can be caused by configuration problems, by defects in the runtime library, or by defects in a program.

If the runtime library detects a configuration problem or a version conflict, the runtime library sends a message to the home terminal of the program and to the EMS event log, commits the outstanding automatic transactions and abends the program.

Likewise, if the runtime library causes a program to trap or if the runtime library detects an irrecoverable internal logic error, the trap message and the stack trace are displayed on the home terminal of the process and in the EMS event log. Always check the home terminal or the EMS event log when program failures occur. See EMS Log below for ways to check for NonStop AutoTMF software events.

Report any program failure that is caused by the NonStop AutoTMF software runtime to HP product support.

Error 75

Error 75 is returned when a “requesting process has no current process transaction identifier”. Since NonStop AutoTMF software is designed to provide transactions whenever they are required, there are a few well-known causes for this error:

1. The monitor process is not running. To check on the state of the monitor process, use the CI command STATUS MONITOR. To start the monitor process, use the START MONITOR command.
2. The NonStop AutoTMF software installation has not been licensed or a time–limited license has expired. Use the MONITOR STATUS LICENSE command to check the state of the NonStop AutoTMF software license.
3. The program has not been prepared. To verify that a program has been prepared successfully, use the CI command INFO PROGRAM. To prepare a program, use the PREPARE command.
4. The configuration disables automatic transactions for the files being accessed or for all files. Use the INFO ATMFFILESET, INFO ATMFPROGRAMS, and INFO GLOBALS commands to verify the configuration parameters.

If the monitor process is running and healthy and the failing program is properly prepared, then the program should be traced to examine the sequence in which the file operations are performed and transactions are started and committed.

Locking Problems

Contention and Deadlocks

Audited files are subject to different locking protocols than non-audited files. When accessing audited files, locking is enforced by TMF to maintain data integrity. An application that was not designed for audited files may incur concurrency problems once locking is dictated by more stringent rules. The table below compares non-audited and audited locking rules:

Table C-1. Non-Audited and Audited Locking Rules:

Non Audited	Audited
Locks are never required by the file system ● Not required for updates or deletes ● Inserts do not create locks	Locks are required for certain operations ● Update or delete require an explicit lock operation
All locking operations are explicit	Locking operations are explicit and implicit ● Implicit for inserted records ● Explicit otherwise
All unlock operations are explicit	Unlock operations are explicit and implicit ● Explicit for unmodified records ● Implicit for modified records when the transaction is committed
Locks are held by a file open and cannot be shared	Locks are held by a transaction and can be shared

NonStop AutoTMF software supplies a LISTLOCKS (LL) command with options to assist in identifying the processes and transactions that are holding locks. Particularly useful are the following:

- The TXSTATUS option displays a summary of all transactions or processes that are holding locks.
- The DEADLOCK option display sets of granted and waiting lock requests that form a deadlock. If the RESOLVE option is specified with DEADLOCK, the command presents the user with a list of transactions and processes that are participants in the deadlock and prompts the user to select either a process to abend or a transaction to abort to resolve the deadlock. Note that deadlocks can resolve themselves if the waiting operations use timed I/O.

Lock collisions, delays and deadlocks typically can be solved by a change in configuration, such as configuring a file as SEPARATETX, changing MAXUPDATE values, setting ISOLATION to NORMAL or STRONG, and so on.

Once the programs in contention are identified, tracing the programs is the surest way to identify the configuration change that will correct the situation.

Long-Running Transactions

Long-running transactions can lead to unilateral aborts if the TMF AutoAbort time limit is reached. Unilateral aborts can have a severe impact on an application because all non-committed updates from a process are rolled back.

Long-running transactions result from locks that are not being released by the process, thus preventing NonStop AutoTMF software from committing the automatic transaction. This situation is often found in cases where:

- Several files are updated under the common transaction and one file or another has a record locked at all times.
- A file managed under a separate transaction has at least one record locked at all times.

If NonStop AutoTMF software attempts to commit an automatic transaction at a point and cannot commit the transaction, an EMS message is sent to warn the operator.

To avoid unilateral aborts due to long-running transactions, configure the AutoCommit time limit as described in Forced Transaction Commit on page 4-17.

Long-running transactions are most often remedied by configuration changes such as configuring a file as SEPARATETX or RECORDTX; however, first you must clearly identify the cause of the long-running transaction by tracing the program.

Caution. If you detect a long-running transaction, do not abort the transaction using TMF-COM. Instead, use the Escort STOP_PROCESS command. An operator abort eventually results in a program failure and the uncommitted updates are lost. STOP_PROCESS commits all outstanding transactions before stopping the process.

Incorrect Behavior

If a prepared program is not returning the correct data or if its behavior has changed unexpectedly, a logic error in the runtime library, a logic error in the application program or a configuration problem may exist.

For example, if a program depends on the value of the file-type parameter returned by a call to FILEINFOREC or GET_FILEINFOLIST, a change in program behavior could occur because auditing the file changes the value returned in the file-type. In this case, set the HIDEAUDIT file attribute to conceal that audit is set for a newly audited file. HIDEAUDIT preserves the value of the file-type.

Otherwise, trace the program before auditing the file and again after auditing the file to identify the source of the behavior change.

NonStop AutoTMF Software Errors

Monitor Process Errors

The monitor is a fault-tolerant process pair designed to be continuously available. All warning and error messages reported by the monitor are displayed on the monitor's home terminal or, if no home terminal has been configured explicitly, to the default EMS collector. You must select a home terminal for the monitor that is well integrated with the application's operational environment and can be monitored along with all other vital subsystems.

The home terminal for the monitor process is configured when the MapDB is created or when the monitor is started. See the description of the CI commands CREATE MAPDB, ALTER MONITOR and START MONITOR for details about configuring the monitor.

The monitor activity can be logged to a file for cases where the monitor's behavior is under suspicion.

If the monitor process fails, the monitor produces a saveabend file and displays a message and a stack trace on its home terminal. Always check messages on the home terminal when a failure occurs.

CI Errors

The CI is the user interface to NonStop AutoTMF software. The CI is used to configure and monitor the environment. The CI also features a series of utility commands such as COPY, FILEINFO, LISTLOCKS (LL), and so forth.

The CI uses the runtime library like any other prepared application program. A defect in the runtime library could therefore affect the CI.

If the CI fails, the CI produces a saveabend file and displays a message and a stack trace on its out file, most often the user's terminal.

Diagnostic Tools

Application Debugging Environment

Debugging programs is an important diagnostic tool. Debugging prepared programs is no different than debugging regular programs. There are only the following minor differences:

- Because Enscribe procedure calls are intercepted by the NonStop AutoTMF runtime, you can set a breakpoint on an Enscribe procedure such as OPEN, READ, WRITE, so on, and step into the procedure code. If this occurs, just step out of the procedure code and resume debugging.
- You can see the open of the TFILE ($TMP) when displaying the files opened by the program.
- If you are tracing the program, you see the open of the trace file when displaying the files opened by the program.

EMS Log

The NonStop AutoTMF software monitor sends two classes of messages to the EMS log:

- Startup and shutdown information
- Critical events

Events triggered by an application failure that is caused by NonStop AutoTMF software are also sent to the EMS log.

Consequently, the first place to look for the cause of a failure in the EMS log.

To quickly view the NonStop AutoTMF events in the EMS log, start a distributor and use the ESCFLTR filter file to select the events. At the TACL prompt, enter the following:

```
ADD DEFINE =_EMS_TEMPLATES, CLASS MAP, FILE ZESCTMPL
EMSDIST TYPE P, COLLECTOR $0, TEXTOUT $HOME, FILTER ESCFLTR
```

You should keep this distributor running in a dedicated window during the installation of NonStop AutoTMF software and during the testing phase.

For example, if the primary monitor process stops, due to a CPU failure for example, the EMS log will show the following messages:

```
09:34 02JUN11 099,02,014 $ZESC 104 ESCMON backup started, cpu 3
09:36 02JUN11 099,03,034 $ZESC 103 ESCMON backup Takeover 1
09:36 02JUN11 099,03,034 $ZESC 112 ESCMON ATMFFILE table cached. Table
                                    entry count = 1.
09:36 02JUN11 099,03,034 $ZESC 112 ESCMON ATMFPROG table cached. Table
                                    entry count = 2.
09:36 02JUN11 099,03,034 $ZESC 101 ESCMON started Version
                                    1.9.1 - 09JUN2011
09:36 02JUN11 099,03,034 $ZESC 104 ESCMON backup started, cpu 2
```

CI Commands

The CI has a rich set of commands to control and monitor the NonStop AutoTMF software environment. This subsection discusses their usage.

Checking the Environment

To check the overall processing environment, use the CI command ENV. ENV displays the names of the current MapDB, SysDB, the Guardian OS release version, the NonStop sever serial number of the system, and so on.

Checking the NonStop AutoTMF Software Configuration

You have options for processing of newly audited files. These options are configured as NonStop AutoTMF software file attributes and program attributes or global parameters. Some attributes can be specified with DEFINEs. If a prepared program does not behave as expected, first check configuration parameters.

To list the values of system wide parameters, use the CI command INFO GLOBALS. To change any global parameter, use ALTER GLOBAL.

To list the NonStop AutoTMF software configuration for selected audited files (SEPARATEX, COMMONTX, and so on), use INFO ATMFFILESET. To configure files, use ADD ATMFFILESET.

To list the NonStop AutoTMF software configuration for selected programs, use INFO ATMFPROGRAMS. To configure programs, use ADD ATMFPROGRAMS.

To verify if a specific DEFINE has been added for a program, look at a trace of the program's execution. All active DEFINEs are listed in the output of the trace.

Checking the State of the Monitor and MapDB

The monitor process must be up at all times. To check on the status of the monitor, use the STATUS MONITOR command to display the monitor's vital statistics.

To look at the configuration of the MapDB, use the INFO MAPDB command.

Checking the State of the Application Programs

A program that has not been prepared will run without warnings until the program accesses an audited file without having an active transaction. The operation will then fail with an error 75. To verify that a program is properly prepared, use the INFO PROGRAM command. To display a list of unprepared programs, issue the command on a file-set using the UNPREPARED option.

To find out who is using the runtime library, use the INFO LIBRARY command. This command displays all the processes that have the library opened, including the CI from which the command is issued.

Checking for Locks

Locking of audited files is different than locking of non-audited files. This could introduce concurrency problems. To display the locks held for a file, a file set, or the

entire system, use the CI LISTLOCKS (LL) (or LL) command. Use the DEADLOCK option of the command to display deadlocks.

Tracing

NonStop AutoTMF software offers a comprehensive set of tracing options. Virtually all file system and TMF operations performed by a program can be traced along with the data that is read and written by the program. Tracing also displays the active DEFINEs, assigns, the startup message, and process termination messages.

Remember that tracing is a diagnostic tool. As such, tracing can generate large amounts of data. Tracing should be used with caution in any production environment and not at all in performance critical environments.

Tracing is initiated in two ways:

- The CI TRACE command.
- DEFINEs

You can trace a single process or all instances of a running program. Up to thirty two such traces, numbered 0 to 31, can be active simultaneously.

To stop tracing, use TRACE STOP.

Static vs. Dynamic Tracing

You can trace a single process or a set of programs. Tracing can be either static or dynamic:

- Static tracing starts at the time the process is launched. Tracing is initiated by the monitor process through the CI TRACE command.

  Tracing can also be started by specifying one of three DEFINES in the program startup command stream or the Pathway server configuration.

- Dynamic tracing can be initiated at any time before or during program execution by the monitor through the CI TRACE command.

  You can enable dynamic tracing on your system with the ALTER GLOBAL command to set the global DYNAMICTRACE to ON.

  The global setting DYNAMICTRACE can be overridden for a single process by using DEFINEs. See =_ESCORT_DYNAMIC_TRC_OFF and =_ESCORT_DYNAMIC_TRC_ON for details.

  Note that the dynamic tracing global or DEFINE must be specified when a process is started, which tells the process to check for tracing at various times.

  Allowing dynamic tracing increases overhead, so use dynamic tracing judiciously.

Tracing and Debugging

Tracing is often done in conjunction with the debugging of a program using INSPECT. In addition to the standard TACL and Pathway DEBUG options, you can specify that a program should be started in INSPECT with the CI TRACE command. Tracing can be particularly useful when debugging Pathway servers since tracing does not require modifying the server configuration.

Trace Files

The trace output can be written to a disk file or terminal. If the disk file specified does not exist, it as created as an entry-sequenced file.

Tracing to an Edit file is more efficient than the alternatives. However, an Edit file can only have one updater, so only one process at a time can be traced to an Edit file.

Some restrictions on tracing may be applied for security reasons. The global parameter SECURETRACE can be used to restrict tracing to the users that own a program. See ALTER GLOBAL on page 6-34 for a description of the SECURETRACE parameter.

Tracing time limit

By default, a trace initiated with a TRACE command will be cancelled after 60 minutes. To change this default, use the FOR option on the TRACE command. To stop the trace before the time limit is reached, simply issue the TRACE STOP command.

Trace Interpretation

The trace is more than a diagnostic tool for finding NonStop AutoTMF software problems; the trace is a useful facility for customers to understand their program’s behavior in general. The trace is, however, an internal tool to support problem analysis by product support specialists. The trace format is not formally documented and is subject to change without notice.

The trace shows a basic timeline of process execution. Most entries have a timestamp, that can be used to measure the relative elapsed time of various operations. Beware, however, using the times as absolute costs of system operations. The tracing activity adds overhead to the process and will lengthen the overall execution time.

The trace shows many, but not all, aspects of process execution. The trace can only show procedures that are intercepted by the NonStop AutoTMF software runtime library, such as most file system and transaction management procedures. Procedures that are not intercepted include process management and memory management.

Most intercepted procedures display a single line of trace, identifying the procedure and the most important arguments.

The trace shows the DEFINES that are active when the process begins execution and the operations to read the startup messages. The STARTUP and ASSIGN messages are formatted and displayed..

```
14:44:03.782(8,102) *** Start Trace ***
14:44:03.823(8,102) Program $PRPC.RWCAWAIT.AWAIT, Term $Z31T
(8,102) DEFINE =CAF,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.CAF
(8,102) DEFINE =CAFKEYS,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.CAFKEYS
(8,102) DEFINE =PBF,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.PBF
(8,102) DEFINE =PRDF,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.PRDF
(8,102) DEFINE =PTDF,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.PTDF
(8,102) DEFINE =PTLF,CLASS MAP,FILE=\FOXII.$PRPC.SOMEDATA.POYYMMDD
(8,102) DEFINE =_DEFAULTS,CLASS DEFAULTS,VOLUME=$PRPC.ESCORT
(8,102) DEFINE =_DEFAULTS,CLASS DEFAULTS,SWAP=\FOXII.$RWC
14:44:04.340(8,102) FILE_OPEN_($RECEIVE:0,RW/SH/NW, Device Type=2)
14:44:04.368(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.388(8,102) System Message: OPEN
14:44:04.411(8,102) AutoTMF $RECEIVE inherited(Null Tx)
14:44:04.414(8,102) AWAITIO($RECEIVE:0,94) Error 6
14:44:04.438(8,102) REPLY(,0,0)
14:44:04.441(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.443(8,102) AutoTMF $RECEIVE inherited(Null Tx)
(8,102) Startup Default \FOXII.$PRPC.ESCORT
(8,102) Startup In File \FOXII.$Z31T
(8,102) Startup Out File \FOXII.$Z31T
14:44:04.448(8,102) AWAITIO($RECEIVE:0,68) Sender: $Z31V
14:44:04.450(8,102) REPLY(,4,0)
14:44:04.452(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.453(8,102) AutoTMF $RECEIVE inherited(Null Tx)
(8,102) ASSIGN PRINT,$S.#REPORT
14:44:04.478(8,102) AWAITIO($RECEIVE:0,108) Sender: $Z31V
14:44:04.498(8,102) REPLY(,0,0)
14:44:04.500(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.503(8,102) AutoTMF $RECEIVE inherited(Null Tx)
(8,102) ASSIGN INFILE,$PRPC.SOMEDATA.POSKEYS
14:44:04.505(8,102) AWAITIO($RECEIVE:0,108) Sender: $Z31V
14:44:04.508(8,102) REPLY(,0,0)
14:44:04.509(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.514(8,102) AutoTMF $RECEIVE inherited(Null Tx)
14:44:04.515(8,102) AWAITIO($RECEIVE:0,25) Sender: $Z31V
14:44:04.517(8,102) REPLY(,0,0)
14:44:04.518(8,102) READUPDATE($RECEIVE:0,0) Nowait
14:44:04.520(8,102) System Message: CLOSE
14:44:04.521(8,102) AutoTMF $RECEIVE inherited(Null Tx)
14:44:04.523(8,102) AWAITIO($RECEIVE:0,4) Error 6
14:44:04.524(8,102) REPLY(,0,0)
14:44:04.526(8,102) CLOSE($RECEIVE:0) Sender: $Z31V
```

When a file is opened, the trace shows the file name, file number, and various open parameters, such as access, exclusivity, nowait, and unstructured access. In addition, the file type is shown. Finally, if the file is audited the NonStop AutoTMF software configuration, either explicit or implicit, is displayed. There are four values for this information:

- ATMF enabled - not configured, but NonStop AutoTMF software will supply automatic transactions if the application does not have a transaction.
- ATMF sep tx - configured for separate automatic transaction.
- ATMF common tx - configured for separate automatic transaction.

- Audited - NonStop AutoTMF software disabled or file not configured for automatic transactions.

```
OPEN($DATA.SOMEDATA.POSKEYS:1,RO/SH, ES)
OPEN($DATA.SOMEDATA.CAF:2,RW/SH, KS, ATMF enabled)
OPEN($DATA.SOMEDATA.PTDF:4,RW/SH, RL, ATMF common tx)
OPEN($DATA.SOMEDATA.PBF:5,RW/SH, KS, ATMF common tx)
OPEN($DATA.SOMEDATA.PRDF:6,RW/SH, KS, ATMF enabled)
OPEN($DATA.SOMEDATA.POYYMMDD:7,WO/SH/NW, ES, ATMF sep tx)
```

NonStop AutoTMF software also keeps track of the locks held by a program on the audited files it manages. The lock count is displayed in the trace when the program performs a locking or unlocking operation under an automatic transaction, as shown below:

```
15:29:37.044($Z40K) READLOCKX($DATA.PROD.ACCT:12,LC=1,CLC=1,310)
...
15:29:37.074($Z40K) WRITEUPDATEUNLOCKX($DATA.PROD.ACCT:12,LC=0,CLC=0,310)
...
15:29:37.389($Z400) UNLOCKFILE($DATA1.PROD.CUST:11,LC=0,CLC=0)
```

There are two lock count values:

- LC - the lock count for the file.
- CLC - the total file locks held under the current automatic transaction.

If a program is experiencing locking problems, long transactions, or errors 35, use the trace to determine why NonStop AutoTMF software does not find a good opportunity to commit automatic transactions.

If a program abends or encounters unexpected errors, trace the program to determine the sequence of procedure calls that lead to the failure.

Problem Reporting

To report a problem please collect supporting data and contact product support. This section lists the most commonly requested information to diagnose a problem. The user may be asked by the support specialist to collect additional information for specific problems.

NonStop AutoTMF Software Component Failures

Prepared Program Incorrect Behavior or Failure

Locking Problems, Long Running Transactions, Errors 35 or Performance Problems

NonStop AutoTMF Software Component Failures

If the monitor process or the CI abends, a saveabend file is produced. The monitor displays a message on its home terminal or the configured EMS collector process and the CI displays a message on its out file.

If ESCMON or ESCORT fail, send the following information:

- A description of the problem, its repeatability, frequency (if applicable), how to recreate the problem, and so on.
- The saveabend file.
- The text of the message on the home terminal or EMS log.
- VPROC output of the failing ESCMON or ESCORT object file.
- The global parameter values output by INFO GLOBALS.

Prepared Program Incorrect Behavior or Failure

If a prepared process produces incorrect results, send the following information:

- A description of the problem, its repeatability, frequency (if applicable), how to recreate the problem, and so on.
- General context information such as whether the application is TMF-aware. If the application is TMF-aware, which files accessed by the failing program were previously audited and which ones are newly audited.
- The text of the message on the home terminal or EMS log.
- If the problem is reproducible, a trace of the program execution.
- The output of INFO ATMFFILESET, INFO ATMFPROGRAMS, and INFO GLOBALS to understand the file and program attributes and global settings.

If a prepared program fails, send the information listed above in addition to:

- The saveabend file (if applicable).
- The object file of the failing program.
- If the NonStop AutoTMF software runtime has been combined with a user library, the object file of the user library.

Locking Problems, Long Running Transactions, Errors 35 or Performance Problems

- The text of the message on the home terminal or EMS log. NonStop AutoTMF software issues warnings when it attempts to commit a transaction and is prevented from doing so for any reason.
- A description of the problem, its repeatability, frequency (if applicable), how to recreate the problem, and so on.
- General context information such as whether the application is TMF-aware. If the application is TMF-aware, which files accessed by the failing program were previously audited and which ones are newly audited.

- The output of LISTLOCKS (LL) at the time of the problem occurred to identify which file is in contention.
- A trace of the programs that access the file in contention.
- The output of INFO ATMFFILESET, INFO ATMFPROGRAMS and INFO GLOBALS to understand the file and program attributes and global settings.

D Error Messages

In each message description, the following information appears:

- Message number
- Message text
- Cause—the condition or error that produced the message
- Effect—the effect of the condition or error on the system
- Recovery—the steps required to recover from a reported error

Informational Messages

100

```
Monitor initialization
```

Cause. The monitor process is initializing itself.

Effect. The monitor process is not ready to service requests until initialization is completed.

Recovery. Informational message only; no corrective action is needed.

101

```
Monitor started
```

Cause. The monitor process has completed initialization.

Effect. The monitor is ready to service requests.

Recovery. Informational message only; no corrective action is needed.

102

```
Monitor stopped
```

Cause. The monitor process is stopping due to a user request. The EMS message identifies the user that made the request.

Effect. The monitor stops.

Recovery. Restart the monitor.

103

```
Monitor takeover
```

Cause. The monitor primary process or its CPU has failed.

Effect. The monitor backup process has taken over. The monitor is ready to service requests.

Effect. Action: Informational message. No action is required.

104

```
Monitor backup created
```

Cause. The monitor primary process has created a backup process.

Effect. The monitor process is fault-tolerant.

Recovery. Informational message. No action is required.

105

```
Monitor backup failed
```

Cause. The monitor backup process has failed.

Effect. The monitor primary process continues operation, but, the monitor process is not fault-tolerant.

Recovery. Determine the cause of the failure. If the cause can be corrected, use the MONITOR BACKUPCPU command to start the backup process.

106

```
Monitor initialization failed
```

Cause. The monitor process failed to initialize itself.

Effect. The monitor process abends. Application processes can still be run, but no services will be provided and processes may fail if the processes require those services.

Recovery. Report the error information to product support.

107

```
Monitor status info
```

Cause. The monitor process has displayed status information.

Effect. Status information is placed in the EMS log.

Recovery. Informational message only; no corrective action is needed.

141

```
Overdue commit on automatic transaction.
```

Cause. A long-running automatic transaction has been detected. NonStop AutoTMF software usually commits transactions after a short interval, but may not be able to do so if applications have long-lasting database locks. NonStop AutoTMF software reports long-running transactions after five minutes.

Effect. The automatic transaction is not committed. NonStop AutoTMF software will continue to attempt committing the transaction. Eventually, if the transaction is not committed by the time the TMF AutoAbort timeout is reached, a unilateral abort will cause the program to fail and the transaction to be backed out.

Recovery. Informational message only. Determine the cause and effect of the long-running transaction. Ensure that the long-running transaction is not subject to an AutoAbort; otherwise, you may need to change the TMF AutoAbort timer.

142

```
MAPDB access error
```

Cause. Configuration could not be accessed from the MapDB tables.

Effect. Configuration information is not available to the monitor; requested services will not be performed.

Recovery. Correct the table access problem and issue the MONITOR REFRESH NOW command.

143

```
MAPDB inconsistent
```

Cause. Configuration information in the MapDB tables is inconsistent.

Effect. The configuration information is not available on request to the monitor.

Recovery. Correct the configuration information.

150

```
AutoTMF message
```

Cause. A message from NonStop AutoTMF software has been generated.

Effect. The nature of the problem is described in the message.

Recovery. Corrective action depends on the message.

Critical Event Messages

402

```
Monitor intentionally abended
```

Cause. The monitor process has been intentionally abended to produce a saveabend file for problem diagnosis.

Effect. The primary monitor process abends. The backup process takes over and continues processing.

Recovery. Send the saveabend file to product support.

403

```
 Invalid license
```

Cause. An operation was attempted, but no license for that service has been installed.

Effect. The operation is rejected.

Recovery. Obtain and install the necessary license.

406

```
 Communication error between the runtime and the monitor
```

Cause. The dialogue between the NonStop AutoTMF software runtime and the monitor failed.

Effect. The program abends.

Recovery. Check for other messages in the EMS log. One probable cause is that the runtime and the monitor are different versions. Correct the problem and restart the program.

407

```
Version mismatch between the monitor and the runtime
```

Cause. The Monitor process and the Runtime have incompatible versions.

Effect. The program abends.

Recovery. Verify that the runtime library and monitor are the same version. Contact product support for further assistance.

410

```
Error reading the object file
```

Cause. The object file is secured to prevent reading by the user ID running the process.

Effect. The program abends.

Recovery. Escort requires access to the object file. Re-secure the object file to allow read access.

411

```
Assertion failure
```

Cause. A logic error has been detected by the process.

Effect. The process abends.

Recovery. Report the error information to product support.

412

```
Resize segment error
```

Cause. An attempt to resize the extended segment failed.

Effect. The process abends.

Recovery. The error number is provided in the message. Determine the reason for the failure and correct it.

413

```
Memory pool allocation error
```

Cause. An attempt to allocate memory in the extended segment failed.

Effect. The process abends.

Recovery. Since the product usually resizes the segment to satisfy memory requests, this error should not occur. Report this error to product support.

414

```
Allocate segment error
```

Cause. Allocation of the extended segment failed.

Effect. The process abends.

Recovery. The error number is provided in the message. Determine the reason for the failure and correct it.

415

```
System operation failed
```

Cause. A system operation needed by NonStop AutoTMF software failed.

Effect. The process abends.

Recovery. Report the error information to product support.

419

```
Utility program process trap
```

Cause. A product component encountered a process trap.

Effect. The process abends.

Recovery. Report the error information to product support.

421

```
A file is not audited
```

Cause. The NonStop AutoTMF software global ATMFABENDNOAUDIT is enabled, the specified file is configured for automatic transactions, but the file is not audited.

Effect. The process abends.

Recovery. Audit the file or change the NonStop AutoTMF software configuration.

422

```
Program calls unsupported SRL procedure
```

Cause. Support for TNS/R native-mode programs required the elimination of a few public SRL procedures because linkable versions of the SRLs were not available.

Effect. The program abends.

Recovery. Report the error information to product support.

428

```
The AutoTMF autocommit time limit has been exceeded
```

Cause. An automatic transaction exceeded the AUTOCOMMIT time limit.

Effect. All automatic transactions were committed and the process has been abended.

Recovery. Determine the cause of the long-running transaction. If a long-running transaction is acceptable, alter the Nonstop AutoTMF software configuration to disable AUTOCOMMIT processing.

429

```
Invalid use of the Record Transaction feature
```

Cause. Record Transactions were configured, but a program performed an operation that could not be completed with this option.

Effect. All automatic transactions were committed and the process has been abended.

Recovery. Determine the nature of the operation and either disable Record Transactions for the program or alter the program to avoid the operation.

430

```
Process was stopped by user request
```

Cause. An authorized user requested a NonStop AutoTMF software process stop.

Effect. All automatic transactions were committed and the process has been abended.

Recovery. Informational message only; no corrective action is needed.

433

```
System contains more logical processors than permitted by
AutoTMF license.
```

Cause. The referenced product is licensed for a maximum number of logical NSK processors (MAXCPUS). The number of active processors exceeds the license maximum.

Effect. Your license is not valid.

Recovery. Contact the license manager requesting a valid license with sufficient number of logical processors. Please provide your system serial number and order number.

600

```
Monitor start unauthorized
```

Cause. An unauthorized user attempted to start the monitor process.

Effect. The monitor process is not started.

Recovery. Start the monitor process with an authorized user ID.

Action Event Messages

701

```
Product object missing
```

Cause. A required program object file is missing from the product subvolume.

Effect. The process abends.

Recovery. Install the missing object file and re-run the program.

704

```
Access to AutoTMF monitor failed
```

Cause. Request to a monitor process returned an error.

Effect. The process abends.

Recovery. Correct the cause of the error displayed in the message and restart the process.

Glossary

automatic transaction. A transaction that has been generated by NonStop AutoTMF software.

common transaction. An automatic transaction that is used for accesses on all file opens that are not configured for separate transactions.

nontransactional program. A program that does not manage or inherit transactions. See transactional program.

prepared program. An object file that has been prepared to use NonStop AutoTMF software.

process transaction. A transaction either started or inherited by a transactional program.

separate transaction. An automatic transaction that is used for accesses on a single file open.

transactional program. A program that is designed to access audited database files and manages transactions explicitly. Also, a server program that inherits transactions from a transactional requestor.

transactional program.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z