Table of Contents

Section 1: IDent Product Overview

IDent implements a number of features which allow MPE sites to significantly enhance their ability to monitor and control access to restricted data, and to create highly secure audit logs. IDent has been tailored to the particular needs of companies faced with becoming compliant with the PCI (Payment Card Industry) standards.

IDent's features are tightly integrated, but may be divided into the following major areas:

- User Keystroke Logging.

  Everything that any user types anywhere can be recorded, and retained for later audit. The audit trail includes whatever prompt was displayed (to simplify interpreting the audit file), a timestamp, and annotations if the user attempted to access any restricted files, or if they boosted their MPE capabilities via GOD or any other capability boosting program.

- TurboIMAGE Database Password Management.

  TurboIMAGE password security can be configured by rules that allow database access to be defined independently of which password a user (or program) supplies. Rules are held external to databases, and allow TurboIMAGE passwords to be changed as often as required, without requiring any changes to the application programs that open the databases. This allows applications to continue working when TurboIMAGE passwords are changed, even if application passwords are hard-coded and the source is unavailable.

- TurboIMAGE Dataset Access Security and Logging.

  Both read and/or write access to particular datasets can be closely controlled, and all accesses can be logged. This feature may be used to closely control and monitor access to datasets containing Credit Card information. Accesses are logged in the same file used for User Keystroke logging, and can show the full audit trail of commands issued by users who gain (or attempt to gain) access to restricted datasets.

- File Access Security and Logging.

  Access to particular files (or filesets) can be closely controlled, and all file accesses can be logged. This feature may be used to closely control and monitor access to files that contain restricted information. Accesses are logged in the same file used for User Keystroke logging, and can show the full audit trail of commands issued by users who gain (or attempt to gain) access to restricted files.

- Critical File Purge Protection.

  Individual files (or filesets) can be protected against being purged or erased. This feature is primarily intended to protect IDent audit files from tampering, and to provide an audit trail of any attempts to delete or modify the files, but may also be used to protect log files used by other products. Protection extends to attempts to purge files by any means (including, for example, PURGE, PURGELINK, PURGEGROUP, PURGEACCT, RESTORE, etc.), and can also guard against tampering by users with SM capability. Purge attempts are logged to the same file used for User

Keystroke logging, and can show the full audit trail of commands issued by users who attempt to purge critical files.

- Detects Modifications to User-Defined 'Critical' System Files.

Critical files may be periodically scanned for modifications. IDent does not rely on timestamps to detect if files have been modified, but instead uses a cryptographically-secure hashing algorithm ('Whirlpool') to calculate a 512-bit message digest (or checksum) for every monitored file. These checksums are stored in a protected location, and can be used by IDent to automatically flag if a monitored file has been added, modified, or deleted. The checksum database can be stored remotely for additional security.

IDent is implemented using a number of features that further assist in implementing enhanced security:

- Central Configuration File with Simple and Flexible Rule Syntax.

IDent uses a powerful and flexible syntax designed to simplify setting up access control rules. IDent's rule-driven features allow rules to be defined that restrict access as a function of many different attributes, including the user's logon (job/session name, LDev number, IP address), the program's environment (program name, the UDC and/or command file used to access the program, and the program's STDIN file name), and advanced attributes such as the names of ancestor processes, and even the names of other files opened by the program. Rule syntax allows lists of authorized Users and Programs to be centrally defined and referred to by synonyms in other rules, simplifying complex configuration needs.

- Ability to Protect Against Attack by SM Capability Users.

IDent protection rules potentially apply to all Users, and the product can be configured to provide substantial protection against unauthorized access by any user, including users with SM capability.

- Flexible Violation Logging.

Access attempts (whether allowed or blocked) can be logged to the same file used to hold each User's keystroke audit trail, and can also be copied to the System Console, to a dedicated Security Monitor Session, and even displayed on the User's own terminal.

- Remote Log File Storage.

The audit files created by IDent may be automatically copied to a remote system (using FTP), as soon as the job or session they refer to logs off. The audit files may also be renamed and archived on the HP3000 system, in a protected location.

- Simple Installation and Removal.

IDent can be enabled or disabled by typing a single command; all configuration commands and access rules are defined in a single ASCII file, and no changes are required to any Jobs, UDCs, Command Files, or Application programs to take full advantage of all the security enhancements that it implements. When IDent is disabled, systems revert immediately to their state prior to enabling IDent, with no lasting effects.

User Input Logging Summary

All interactive commands and data typed by any User into any program (including MPE's ':' prompt) may be logged. The ability is provided to selectively specify which users' input must be logged; users may be specified as a function of their logon ID, logon IP address or LDev number, or logon capabilities.

IDent is configured using a powerful and flexible configuration file syntax. You can, for example, log everything typed by a particular user, and also log all commands executed in Jobs that the user STREAMs, regardless of how those Jobs log on.

IDent User Log files are stored in a secure group on the HP3000; they may automatically be sent by FTP to a remote system as soon as the monitored User logs off.

IDent User logging may be configured to log all User input, and to automatically discard the Log file when the User logs off unless the User attempted to access 'protected' files while logged on. When configured this way, the resulting log file contains all commands and data typed by the user prior to their attempt to access the 'protected' file, which may be very useful in determining how access was gained.

It's also possible to configure IDent to only start logging User input if a User boosts their MPE capabilities (using any of the available 'GOD' programs), or if they change their logon (using a logon-switcher such as MPEX's %CHLOGON command) to operate as a higher-capability user.

IDent User Log files contain everything typed by a User, together with the associated on-screen prompt (to simplify analyzing Log files later on). If users type passwords with local echo suppressed, they will be logged as a series of '*' characters, one for every character input.

IDent does not give users any indication that their session is being monitored. It does not set JCWs, CI variables or file equations within sessions being monitored. It only writes to the system console if exceptional internal errors are detected, or if you have configured IDent to send messages to the console.

User Input Logging now also tracks User input typed into block-mode programs and full-screen editors.

IDent's Database and File access features may be configured to log attempts (successful or not) to access particular files and datasets by writing messages to the IDent Log file.

Database Password Management Summary

IDent allows TurboIMAGE database passwords to be changed without also having to modify passwords embedded within application programs. This is made possible using a powerful and flexible configuration file syntax that allows you to specify which database access should be granted to which users running which programs.

You can specify which Database class should be granted as a function of a combination of the database name, program name, user logon, specified database password, user logon 'mode' (Job,

Session, or Job streamed from a qualifying Session). It's also possible to control access as a function of the name of UDC or Command File that was used to launch the application program.

TurboIMAGE Dataset Access Security and Logging Summary

IDent allows read and/or write access to individual datasets to be controlled and/or logged. This feature may be used, for example, to closely control and monitor access to datasets containing Credit Card information.

You can specify which users and programs should be subject to monitoring, or exempt from monitoring, using the full range of IDent selection keywords. These allow access to be controlled as a function of the database name, dataset number, user logon, user logon 'mode' (Job, Session, or Job streamed from within a qualifying Session). It's even possible to control access as a function of the name of UDC or Command File used to launch the application program.

IDent can log Dataset access attempts by writing messages to the IDent Log file, to the System Console, to a Security Monitor Session, and even to the terminal of the User attempting the access.

IDent's User Input Logging may be configured so that the Log file containing all User input is only retained if Users attempt (successfully or not) to access particular TurboIMAGE datasets.

File Access Security and Logging, including Purge Protection Summary

IDent allows individual files or filesets to be protected against being purged or overwritten, regardless of the capabilities of the user. This feature is primarily intended to allow audit log files to be protected against tampering by any user, including users with SM capability. Protected files may still be appended to.

IDent will block attempts to purge protected files using MPE commands, Posix shell commands, and programmatically using calls to documented intrinsics and C/Posix library functions. Protection extends to blocking the :PURGEGROUP and :PURGEACCT commands from removing groups that contain protected files, to blocking use of the FSCHECK program (which can purge files using a special mechanism), and to blocking attempts to purge files using Vesoft's MPEX utility. Protection rules may include exceptions that allow particular users to purge files, limited by User logon, logon LDev, or IP address.

IDent can log file access attempts by writing messages to the IDent Log file, to the System Console, to a Security Monitor Session, and even to the terminal of the User attempting the access.

IDent may also be configured so that the Log file containing a particular User's input is only retained if that User attempted (successfully or not) to access particular files.

Detects Modifications to User-Defined 'Critical' Files.

Any number of files (or filesets) may be verified periodically. IDent uses a high-performance cryptographically-secure hashing algorithm ('Whirlpool') to calculate 512-bit checksums for every monitored file. These checksums are stored in a protected location, and can be used by IDent to automatically flag if any monitored file has been added, modified, or deleted, without having to rely on file label timestamps. The checksum database can be stored remotely for additional security.

The Checksum code can scan any MPE file type (including privileged files like databases); the checksum is calculated as a function of all data within a file (including any user file labels). The checksum algorithm ignores regions within CM programs and libraries that are modified by the loader when programs are run, so that the act of running a CM program will not change a file's checksum. IDent also excludes areas within the NM system library NL.PUB.SYS that are dynamically modified while a system is running. This results in reliable checksums that guarantee that any file modification will be detected, while minimizing false-positives. The checksum scanner can safely be used on MSG and FIFO files (it will not block while reading empty files, and does not destructively read data); it also does not update file last-accessed timestamps when opening files for scanning.

IDent uses the public domain Whirlpool algorithm. Whirlpool is a start-of-the-art one-way cryptographic hash function, adopted by the ISO and IEC. It returns a 512-bit message digest, calculated as a function of every byte within a file. IDent stores the entire message digest in an index file, and uses this to detect file changes without having to rely on file label timestamps.

IDent can also display the calculated 512-bit message digest formatted using hex or base-64 encoding, allowing you to download the checksums and manage them using other mechanisms.

Limitations

IDent limits File Access rules to files whose names can be expressed in traditional MPE syntax; it does not allow files with HFS (Posix) syntax filenames to be protected. IDent will correctly control attempts to access MPE files that are referenced using HFS syntax.

The File Access Log functions can log all file access attempts that are blocked or allowed by the IDent product; it is not designed to log failed file access attempts that are stopped by normal MPE security checks.

Demonstration versions of IDent impose a time limit (normally 20 days from installation), and a limit on the number of lines logged per session (normally 50). Aside from these limitations, demonstration versions function exactly as purchased versions of the product do.

Liability Disclaimer

IDent File protection guards against all known documented file access methods. Users with PM capability and enough knowledge of system internals could conceivably bypass the protection supplied by IDent (as well as any other system protection mechanism).

While every effort has been taken to ensure that IDent provides comprehensive and complete security coverage, complete System Security requires that careful attention is paid to file system and user account security configuration. Consequently the author of IDent expressly disclaims liability for security breaches that occur while using IDent.

Supported Environments

IDent is supported on MPE/iX 6.5 and all subsequent releases.

Support

For support, please contact the author at:

email: ptaffel@gmail.com
phone: 424 442 9003

Section 2: IDent Product Installation

Follow these steps:

1	Transfer the supplied Reflection labels-format file to your HP3000 as IDENT500

2	Restore all files into the TEMP.PTC group:
	
	`:HELLO MANAGER.SYS`
	`:FILE IDENT500;DEV=DISC`
	`:RESTORE *IDENT500; @.@.@; CREATE; SHOW`
	
	All files are restored into the TEMP group of the PTC account.

3	If updating an existing IDent installation, Disable the product while logged on as an authorized user:
	
	`:HELLO MANAGER.SYS`
	`:RUN OFF.PUB.PTC`
	
	Note that your local configuration settings may impose additional restrictions on who is able to Disable the product.

4	Ensure that the PTC account is correctly set up, and move files into their final locations: `:STREAM INSTALL.TEMP.PTC`
5	Password-protect the PTC account, if not already done: `:ALTACCT PTC; PASS=<password>`

NOTE: IDent is not automatically enabled following installation.
See next page for details on how to enable IDent.

Enabling IDent

IDent must be enabled once after installation and following each system restart before its security features become active. IDent is enabled by typing the following:

```
:RUN ON.PUB.PTC
```

You must be logged on as a user with SM or OP capability in order to issue this command.

Running ON.PUB.PTC when IDent is already enabled is not an error; the status message displayed will indicate if IDent was already enabled.

The **Special Protection for Enabling & Disabling IDent** discussion in the **Advanced Topics** section contains information on special protection for the ON.PUB.PTC program.

If automatic FTP exporting of log files has been configured (see the **IDent Log File Handling** section for more information), the background FTPMGR Job will be started automatically when this program is run.

Disabling IDent

IDent may be disabled by typing the following command:

```
:RUN OFF.PUB.PTC
```

You must be logged on with SM capability to issue this command.

Disabling IDent will not affect sessions that are already logged on.

Running OFF.PUB.PTC when IDent is already disabled is not an error; the status message displayed will indicate if IDent was already disabled.

The **Special Protection for Enabling & Disabling IDent** discussion in the **Advanced Topics** section contains information on special protection for the OFF.PUB.PTC program.

If automatic exporting of log files has been configured (see the **IDent Log File Handling** section for more information), the background FTPMGR Job will be shut down automatically when this program is run.

Section 3: Configuring IDent: the CONFIG File

All configuration rules are saved in the file CONFIG.PUB.PTC. This is a flat ASCII file that may be modified using any editor; it should be saved unnumbered in fixed or variable record format. Changes to the CONFIG file normally take effect when users next log on.

The CONFIG file is scanned when every Job or Session logs on, and is also scanned the first time that individual programs attempt to access protected files.

The CONFIG file starts with a number of '#' prefixed lines that contain license-related information. You should only modify the contents of these lines if instructed to do.

The rules for the various components of IDent are located within the CONFIG file, grouped into sections. Each section is prefixed by a '$' section identifier.

CONFIG File Section Identifiers

The following '$' section identifiers can appear in the CONFIG file:

$CONFIG	Defines start of Global Configuration settings
$USERSET	Defines start of User-Set definitions
$PROGSET	Defines start of Program-Set definitions
$LOG-USER	Defines start of User Input Logging rules
$DB-PASS	Defines start of TurboIMAGE Database Password management rules
$DB-ACCESS	Defines start of TurboIMAGE Dataset Access and Logging rules
$FILE-ACCESS	Defines start of File Access, Purge Protection, and Logging rules
$CHECKSUM	Defines start of critical file definitions

CONFIG File Section Summary

Within the CONFIG file, individual configuration settings are located following their associated '$' identifier. The configuration setting syntax varies by section:

$CONFIG	This section consists of a simple list of keywords that globally modify the behavior of IDent. Each keyword controls one aspect of IDent. Some keywords must be followed by associated data values. When the $CONFIG section is scanned, all specified keywords will be recognized; blank lines are not significant in this section.
$USERSET	This section consists of definitions of named lists of User logons. These names may be used by rules in other CONFIG file sections, and allow rules that refer to large lists of Users to be expressed concisely. Blank lines are used to signal the end of each $USERSET definition. When the $USERSET section is processed, scanning does not stop when the first definition matches. This allows a User logon to be a member of multiple User-Set definitions.
$PROGSET	This section consists of definitions of named lists of Program filenames. These names may be used by rules in other CONFIG file sections, and allow rules that refer to large lists of Programs to be expressed concisely. Blank lines are used to signal the end of each $PROGSET definition. When the $PROGSET section is processed, scanning does not stop when the first definition matches. This allows a program to be a member of multiple Program-Set definitions.
$LOG-USER	This section consists of rules that define which User logons should be enabled for input logging. Each rule consists of one or more "Keyword=value" expressions; if a rule contains more than one keyword, all keywords expressions must be True for the rule to take effect. In other words, the individual Keywords in a rule are AND'ed together. Blank lines are used to signal the end of each $LOG-USER rule. When the $LOG-USER section is processed, scanning stops when the first rule matches.

$DB-PASS	This section consists of rules that define which TurboIMAGE Database Passwords should be substituted at run-time. Each rule consists of one or more "Keyword=value" expressions; if a rule is made up of more than one keyword, all keyword expressions must be True for the rule to take effect. In other words, the individual Keywords in a rule are AND'ed together. Blank lines are used to signal the end of each $DB-PASS rule. When the $DB-PASS section is processed, scanning stops when the first rule matches.
$DB-ACCESS	This section contains rules that define which TurboIMAGE Datasets require special access processing to restrict access, and whether these attempts should be logged. Each rule consists of one 'BASE=' keyword, followed by one or more 'Keyword=value' expressions; if a rule is made up of more than one keyword, all keyword expressions must be True for the rule to take effect. In other words, the individual Keywords in a rule are AND'ed together. Blank lines are used to signal the end of each $DB-ACCESS rule. $DB-ACCESS rules are scanned once the first time a process opens any Dataset.
$FILE-ACCESS	This section contains rules that define which files require special access processing to restrict file access and purge attempts, and whether these attempts should be logged. Each rule consists of one 'FILE=' keyword, followed by one or more 'Keyword=value' expressions; if a rule is made up of more than one keyword, all keyword expressions must be True for the rule to take effect. In other words, the individual Keywords in a rule are AND'ed together. Blank lines are used to signal the end of each $FILE-ACCESS rule. $FILE-ACCESS rules are scanned once when a process starts up.
$CHECKSUM	The section contains fileset definitions that define which files are regarded as 'critical'. By default, IDent recalculates checksums for all filesets defined in this section when the supplied CHECKSUM program is run, and can identify new, modified, and deleted files. Each definition consists of one 'FILE=' keyword, followed by a single filename or fileset. Filename may be specified in MPE or HFS (Posix) syntax. A blank line is required after each file definition. $CHECKSUM definitions are scanned when the CHECKSUM program is run.

Configuration Rules Syntax

Within sections that support the use of "Keyword=value" expressions, Rules consist of one or more Keywords. A Rule evaluates True if all its Keywords evaluate True.

Each Keyword may be followed by one or more data values, one per line. For a Keyword to evaluate True, at least one of the specified data values must match.

In other words, when evaluating if a Rule should be applied, all specified Keywords are AND'ed together, and within each Keyword all associated data values are OR'd together.

The following example shows a rule that controls database Password substitution.

```
$DB-PASS

     BASE=DB.GROUP.ACCT
          @.db.sys
     PROG=@.prog.APPS
     USER=JOHN.@:S
          MARY.@:J
          STEVE.APPS
           MANAGER.SYS
     NEWPASS="Secret"
```

This rule is interpreted as:

```
IF  (   Base is "DB.GROUP.ACCT"
     OR Base is "@.DB.SYS"
    )

AND (   Prog is "@.PROG.APPS")

AND (   User is Session "JOHN.@"
     OR User is Job "MARY.@"
     OR User is "STEVE.APPS"
     OR User is "MANAGER.SYS"
    )

THEN
     NewPass is "Secret"
```

The rule is terminated by a blank line.

Commenting out Parts of the CONFIG File

Lines that start with a '#' character are treated as comments, and ignored. A line starting with a '#' does not signal the end of a rule (as a blank line does), so you can use '#' to "comment out" individual lines within a rule.

The following example shows a rule that controls database Password substitution.

```
$DB-PASS

     BASE=DB.GROUP.ACCT          # First Keyword
          @.db.sys
     USER=JOHN.@:S               # Second Keyword
          MARY.@:J
          STEVE.APPS
   #      MANAGER.SYS
     NEWPASS="Secret"            # New Password
```

The "USER=MANAGER.SYS" line is commented out by the leading '#' character. The rule is terminated by the first blank line.

Trailing comments may also be placed on any line, by appending the comment after a '#' character (as shown above). If the '#' is not the first non-blank character in a line, it must be preceded by at least one blank. Everything starting with the '#' character will be ignored.

When CONFIG File Changes Take Effect

When a Job or Session first logs on, the CONFIG file is scanned, and a local temporary file is created that contains a copy of all rules that apply to the current Job or Session. This is done by evaluating all USER= and TERM= keywords, and then ignoring rules where these keywords evaluate False. This optimizes subsequent CONFIG rule lookup operations, as this 'local' CONFIG file is normally far smaller than the system-wide CONFIG.PUB.PTC file.

As the 'local' temporary CONFIG file is created when Jobs and Sessions first log on, these Jobs and Sessions are then insulated from any changes that are made to the CONFIG file after they log on. Hence any changes made to the CONFIG file are not seen by currently logged-on Jobs and Sessions until they next log on.

See the **Rebuilding a Local CONFIG File** discussion in the **Advanced Topics** section for a description on how to force a local CONFIG file to be rebuilt.

Section 4: CONFIG File $CONFIG Section

Many aspects of IDent behavior may be controlled by keywords in the $CONFIG section, with one keyword per line. Blank lines are not significant in the $CONFIG section, and keywords (some of which are followed by '=' and associated data) are not case-sensitive.

Supported $CONFIG section keywords are described below, grouped logically by function.

PEUTIL Utility Access Control Keywords

PEUTIL-BLOCK	Blocks attempts to use PEUTIL.PUB.SYS to turn off the global AIF:PE (Procedure Exits) facility used by IDent.
PEUTIL-ALLOW-SM	Allows SM users to override the PEUTIL-BLOCK keyword.
PEUTIL-LOG	Logs attempts to use PEUTIL.PUB.SYS; these attempts cause Log files to be retained if LOG-DISCARD is also specified.
	See **Restricting Access to PEUTIL** in the **Advanced Topics** section for more details.

PURGEGROUP & PURGEACCT Command Access Control Keywords

PURGEX-BLOCK	Blocks attempts to use :PURGEGROUP or :PURGEACCT on MPE groups that contain protected files.
PURGEX-ALLOW-SM	Allows SM users to override the PURGEX-BLOCK keyword.
PURGEX-LOG	Logs attempts to use :PURGEGROUP or :PURGEACCT on MPE groups that contain protected files; these attempts cause Log files to be retained if LOG-DISCARD is also specified.
	See **Restricting Access to :PURGEGROUP and :PURGEACCT** in the **Advanced Topics** section for more details.

PTC Account Protection Keywords

ALTX-BLOCK	Blocks attempts to use :ALTGROUP, :ALTUSER, or :ALTACCT commands to alter PTC Account attributes.
ALTX-ALLOW-SM	Allows SM users to override the ALTX-BLOCK keyword.
ALTX-LOG	Logs attempts to alter PTC Account attributes; these attempts cause Log files to be retained if LOG-DISCARD is also specified.

FSCHECK Access Control Keywords

FSCHECK-BLOCK	Blocks attempts to run FSCHECK. Running FSCHECK is allowed if FSCHECK-BLOCK is omitted.
FSCHECK-ALLOW-SM	Allows SM users to override the FSCHECK-BLOCK keyword.
FSCHECK-LOG	Logs attempts to run FSCHECK; these attempts cause Log files to be retained if LOG-DISCARD is also specified.
	See **Restricting Access to FSCHECK** in the **Advanced Topics** section for more details.

:Debug Access Control Keywords

DEBUG-BLOCK	Blocks any attempt to enter the System Debug Facility. Equivalent to specifying both DEBUG-BLOCK-ONLINE and DEBUG-BLOCK-PROGRAM keywords.
DEBUG-BLOCK-ONLINE	Blocks attempts to enter Debug in interactive mode.
DEBUG-BLOCK-PROGRAM	Blocks attempts to enter Debug to execute predefined command sequences programmatically.
DEBUG-ALLOW-SM	Allows SM users to override the DEBUG-BLOCK-ONLINE keyword.
DEBUG-LOG	Logs attempts to enter DEBUG; these attempts cause Log files to be retained if LOG-DISCARD is also specified.
	See **Restricting Access to :DEBUG** in the **Advanced Topics** section for more details.
	See also **LOG-DEBUG-COMMAND** keyword, described below.

"Access-Blocked" Message Destination Keywords

BLOCK-TO-CONSOLE	Send "Access-blocked" messages to the System Console.
BLOCK-TO-LOG	Write "Access-blocked" messages to Job/Session IDent Log file.
BLOCK-TO-MONITOR	Send "Access-blocked" messages to the IDent Security Monitor session.
BLOCK-TO-TERM	Send "Access-blocked" messages to User Terminal.
BLOCK-TO-TERM-FKEYS	Writes "Access-blocked" messages over User Terminal function keys.
	See the **Logging Options** section for more details on the BLOCK-TO-XXX keywords.

"Access-Allowed" Message Destination Keywords

ALLOW-TO-CONSOLE	Send "Access-allowed" messages to the System Console.
ALLOW-TO-LOG	Write "Access-allowed" messages to Job/Session IDent Log file.
ALLOW-TO-MONITOR	Send "Access-allowed" messages to the IDent Security Monitor session.
ALLOW-TO-TERM	Send "Access-allowed" messages to User Terminal.
ALLOW-TO-TERM-FKEYS	Writes "Access-allowed" messages over User Terminal function keys.
	See the **Logging Options** section for more details on the ALLOW-TO-XXX keywords.

$LOG-USER Section Default Setting Keywords

[NO]LOG-DISCARD	LOG-DISCARD discards IDent Log files when monitored Users log off, unless IDent also logged at least one IDent "Access-allowed" or "Access-blocked" message to the Log file. By default (or if NOLOG-DISCARD is specified), IDent Log files are retained even if no Access messages were logged.
[NO]LOG-PRIV-BOOST	LOG-PRIV-BOOST generates a Log message if the User acquires Privileged capabilities by running a 'GOD' program. When LOG-DISCARD is also specified this Log message will prevent the IDent Log file from being discarded. By default (or if NOLOG-PRIV-BOOST is specified), no message is logged when a 'GOD' program is run.
[NO]LOG-PRIV-LOGON	LOG-PRIV-LOGON generates a Log message if a Privileged User logs on. When LOG-DISCARD is also specified this Log message will prevent the IDent Log file from being discarded. By default (or if NOLOG-PRIV-LOGON is specified), no message is logged when a Privileged logon occurs.
[NO]LOG-STREAM	LOG-STREAM generates a Log message if Users Stream a job. When LOG-DISCARD is also specified this Log message will prevent the IDent Log file from being discarded. By default (or if NOLOG-STREAM is specified), no message is logged when a Job is Streamed.

[NO]LOG-PRIV-ONLY	LOG-PRIV-ONLY only logs User input while Users are Privileged. By default (or if NOLOG-PRIV-ONLY is specified), User input is logged regardless of User capabilities.
[NO]LOG-PRIV-NOSTOP	LOG-PRIV-NOSTOP starts logging User input when User gain Privileged capabilities, and keeps logging from that point on. By default (or if NOLOG-PRIV-NOSTOP is specified), User input is logged regardless of User capabilities.
	This group of LOG keywords establish defaults that apply to all $LOG-USER section rules. Individual $LOG-USER rules may override these settings. See the **CONFIG File $LOG-USER Section** for more details.

Log File FTP Transfer Keywords

FTP-ADDR= ***"address"***	Specifies the remote FTP System Name or IP Address.
FTP-USER= ***"user-name"***	Specifies the remote FTP User logon. *user-name* is not automatically up-shifted.
FTP-PASS= ***"password"***	Specifies the remote FTP User password. *password* is not automatically up-shifted.
FTP-PROG= ***"program"***	Specifies the location of the FTP client program to be run. Defaults to "FTP.ARPA.SYS".
FTP-TARGET= ***"pattern"***	Specifies the remote filename. Defaults to "@".
FTP-CMD1= ***"command"***	Specifies the first FTP command sent after logon (optional).
FTP-CMD2= ***"command"***	Specifies the second FTP command sent after logon (optional).
FTP-JOB= ***"job-file"***	Specifies name of file containing FTP background Job (optional).
FTP-START	Specifies that the FTPMGR Job is automatically started or stopped whenever IDent is enabled or disabled.
FTP-TRACE	Displays Console message when a Log file is sent via FTP.
	See **FTP Configuration** in the **IDent Log File Handling** section for more details on FTP configuration.

Logging Keywords

LOG-ARCHIVE="pattern"	Moves IDent Log files from the LOG group to the ARCHIVE group with a user-specified name. See **$CONFIG Section LOG-ARCHIVE keyword** in the **IDent Log File Handling** section for details of pattern syntax.
NOLOG-BLOCK-MODE	By default, when user input is being logged, all user input made into VPLUS screens or full-screen pseudo-block mode applications will also be logged. When NOLOG-BLOCK-MODE is specified, user input into VPLUS screens or full-screen pseudo-block-mode applications will not be logged.
LOG-CHLOGON	Logs all calls to AIFCHANGELOGON (used by MPEX %CHLOGON command, FTPSRVR, and other programs). See **$CONFIG Section LOG-CHLOGON Keyword** in the **Advanced Topics** section for more details.
LOG-CIERROR	Logs all displayed CIERROR and CIWARN messages.
LOG-DBOPEN	Logs all successful and unsuccessful DBOPEN calls. Successful calls identify the User Class granted; unsuccessful calls log the TurboIMAGE error.
LOG-DEBUG-COMMAND	Logs the actual command(s) passed to Debug for execution.
NOLOG-HELLO-PARM	By default, a Log message is generated if an SM User logs on and specifies ;PARM=-1 or -2 on the :HELLO command. This Log message will prevent the IDent Log file from being discarded. The Log message is not generated when NOLOG-HELLO-PARM is specified.
LOG-PCLINK	Logs all files transferred using Reflection's file-transfer feature. See **Logging Reflection File Transfers** in the **Logging Options** Section for more details.
LOG-TRACE	Displays a Console message when a Log file is moved from the LOG group to the ARCHIVE group, or if a Log file is discarded because LOG-DISCARD was specified.
LOG-VIA	Modifies all "Access-allowed" and "Access-blocked" messages written to IDent Log files to also identify the Program, Command File, and/or UDC used in the access attempt.
CC-MASK	Masks all but the last-4 digits of Credit Card numbers written to IDent log files. Includes the optional 2-digit Credit Card type prefix ('VI', 'MC', 'DI', or 'AX').

CC-MASK-ALL	Masks all digits of Credit Card numbers written to IDent log files. Includes the optional 2-digit Credit Card type prefix ('VI', 'MC', 'DI', or 'AX').

Miscellaneous Keywords

OP-IS-PRIV	Controls if OP capability is treated as Privileged when evaluating $LOG-USER section rules. By default, OP capability is not treated as Privileged.
MONITOR-ACCT="acct"	Changes the IDent Monitor logon from PTC to another account.
STATUS-TO-MONITOR	Writes IDent status messages to the IDent Security Monitor session. See **IDent Security Monitor Session** in the **Logging Options** section for more details.
TRACE-RULES	Writes trace messages to a separate IDent trace file as each CONFIG file rule is evaluated. The trace file shows how each rule is evaluated. See **$CONFIG Section TRACE-RULES Keyword** in the **Advanced Topics** section for more details.
TRACE-DBOPEN	Writes information describing the environment for every intercepted DBOPEN call to a separate IDent trace file. The trace information shows the password originally supplied by the calling program, and the password substituted by IDent. See **$CONFIG Section TRACE-DBOPEN Keyword** in the **Advanced Topics** section for more details.

Section 5: CONFIG File $USERSET Section

The $USERSET section allows you to define User-Set names that refer to lists of Users. User-Set names can be used in other CONFIG file sections as a short-hand to simplify rules that use "USER=" keywords to match against many User names.

A User-Set definition consists of the User-Set name, followed by an '=', followed by one or more User names, and terminated by a blank line. Only one User name should be specified on each line. The User-Set name may consist of up to 16 alpha-numeric characters, including '-' and '_' characters, MPE wildcards, and may also contain an optional leading Job/Session name. User-Set names may not include trailing MPE Group names, and are not case-sensitive.

User name specifications may also specify the 'logon context', which allows the User Name to be limited to Jobs or Sessions. The 'logon context' of a User name is specified by appending a ':' followed by one or more of the qualifiers 'J', 'S', 'I', or '@':

:S	Limit context to matching Sessions.
:J	Limit context to matching Jobs.
:I	Limit context to all Jobs streamed by matching Sessions.
:JS	Limit context to matching Jobs & Sessions. This is the default context.
:JSI	Limit context to matching Jobs and Sessions, and to all Jobs streamed by matching Sessions.
:@	Same as :JSI

Example:

```
$USERSET

   Admin-Users=
      Joe,mgr.@:S
      SUSAN,AB@.@

   SYSTEM-Manager=
      MANAGER.SYS:JS
      @,MGR.@
      Operator.Sys

   Online-Users =
      @.@:S

   Batch-Users =
      @.@:J
```

There is no limit to the number of $USERSET section User-Set definitions, and no limit to the number of User names contained within each definition. When the User-Set name is used inside one of the other CONFIG file sections, the User-Set name evaluates True if the active User name matches any of the User names in its User-Set definition. User-Set names are identified in other rules by prefixing them with a '^' character. For an example, refer to the description of the $DB-PASS section below.

A given User name can be a member of up to 16 different $USERSET User-Set definitions; if any User Name matches more than 16 definitions, subsequent definitions will be ignored.

The $USERSET section is scanned once, every time any process is started.

Section 6: CONFIG File $PROGSET Section

The $PROGSET section allows you to define Program-Set names that refer to lists of Programs. Program-Set names can be used in other CONFIG file sections as a short-hand to simplify rules that use the "PROG=" keyword to match against many Programs names.

A Program-Set definition consists of the Program-Set name, followed by an '=', followed by one or more Program filenames, and terminated by a blank line. Only one Program filename should be specified on each line. The Program-Set name may consist of up to 16 alpha-numeric characters, including '-' and '_' characters, and MPE wildcards, but may not contain HFS-format (POSIX-style) filenames. $PROGSET entries are not case-sensitive.

Program filenames may also refer to ancestors of the active program, by specifying the 'generation' of the father process. The process generation is specified by appending a ':' followed by the generation number. The suffix ':0' refers to the active program, ':1' refers to the father of the current program, ':2' refers to the grandfather of the current program, and so on.

By default, all Program filenames refer to the current program (the ':0' suffix is assumed if no suffix is present). For example:

```
$PROGSET

    System-progs=
       @.@.SYS
       @.PUB.TELESUP
       M[a-z]@.PUB.VESOFT

    Apps=
       AP@.PROG.APP@
       @.PUB.DEV

    MENU-SONS =
       MENUPROG.PUB.APPS:1
```

The MENU-SONS Program-Set in the above example will qualify all programs run as direct sons of the program MENUPROG.PUB.APPS.

There is no limit to the number of $PROGSET section Program-Set definitions, nor on the number of Program names contained within each definition. When a Program-Set name is used inside one of the other CONFIG file sections, the Program-Set name evaluates True if any of the filenames within its Program-Set definition match. Program-Set names are identified in other rules by prefixing them with a '^' character. For an example, refer to the description of the $DB-PASS section below.

Only the first 16 matching $PROGSET Program-Set definitions are tracked; if more than 16 Program-Set definitions match, subsequent definitions will be ignored.

The $PROGSET section is scanned once, every time any process is started.

Section 7: CONFIG File $DB-PASS Section

The $DB-PASS section allows TurboIMAGE Passwords or Password Classes to be automatically assigned to specified Programs as a function of how the Program is being run.

The $DB-PASS section can contains any number of Password Rules. When any program calls DBOPEN to attempt to open a TurboIMAGE database, the list of Password rules is scanned for the first rule that matches. If a matching rule is found, the TurboIMAGE Password that is supplied by the calling program will be replaced by the new Password specified in the rule. Rules can specify the User Class number to be assigned, so passwords do not have to be specified in clear text in the CONFIG file.

Individual $DBPASS Password Rules may contain any number of keywords. Every specified keyword must evaluate True for the Rule to match. If any keyword evaluates False, the entire Password Rule will be skipped, and the next $DB-PASS section Password Rule will be evaluated. If no rules match, or if the matching rule has no NEWPASS keyword, then the password supplied by the Program will be used.

Each $DB-PASS section Rule may contain one or more of the following keywords which together limit the scope of the rule:

BASE=	Limits the rule to the specified Database(s). May specify one or more Database names, including wildcards.
CMDFILE=	Limits the rule to programs launched by the specified MPE Command File(s). May specify one or more Command File Names, including wildcards.
MODE=	Limits the rule to DBOPEN calls with the specified Mode number(s). May specify one or more Mode numbers, and/or the letters 'R' and 'W', signifying 'Read' or 'Write' access. Separate multiple numbers with ','s.
OPENFILE=	Limits the rule to programs that have the specified File(s) open. May specify one or more File names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific file(s) open, by appending a '*:n*' generation number suffix to the file name specification, as described in the PROG= keyword description below.
PASS=	Limits the rule to DBOPEN calls with the specified Database Password(s). May specify one or more Database Passwords. Passwords should be quoted if they contain any non-alphanumeric characters or if they contain any lowercase letters (unquoted strings are upshifted, quoted strings are not).

PROG=	Limits the rule to the specified Program(s). May specify one or more Program names, including wildcards. May also refer to $PROGSET section Program-Sets by prefixing Program-Set names with a '^'. This keyword may be used limit the rule to programs with specific parent or 'ancestor' processes by appending '**:***n*', where a '*n*' is the generation number of the ancestor. ':0' refers to the current process, ':1' refers to its 'father', and ':2' refers to its 'grandfather', etc. The default generation suffix is ':0'. You cannot specify ':*n*' process generation suffixes with '^' prefixed Program-Set names (although Program-Set definitions can themselves contain generation suffixes inside the $PROGSET section).
STDIN=	Limits the rule to programs that have the specified STDIN file(s) open. May specify one or more file names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific STDIN file(s) open, by appending a '**:***n*' generation number suffix to the file name specification, as described in the PROG= keyword description above.
TERM=	Limits the rule to users logged on the specified LDev number(s) or IP address(es). May specify one or more LDev numbers and/or IP Addresses, including the MPE '#' or '@' wildcards. Simple numbers are interpreted as LDev numbers; fields that contain embedded '.'s are interpreted as IP Addresses.
UDC=	Limits the rule to programs launched by the specified MPE UDC(s). May specify one or more UDC names, including wildcards.
USER=	Limits the rule to the specified User(s). May contain one or more User names, including wildcards. User names specify the MPE User and Account name, and may contain an optional leading Job/Session name (with a ',' delimiter). May refer to $USERSET section User-Set definitions by prefixing the User-Set name with a '^'. User name specifications may include a trailing ':' and any of 'J', 'S', 'I', or '@'. ':J' limits the context to Jobs, ':S' limits the context to Sessions, ':I' (for 'Indirect') limits the context to any Job streamed by the specified User. ':@' is equivalent to specifying ':JSI'; the default USER= context qualifier is ':JS'.

$DB-PASS section rules may contain the following (optional) keyword which specifies the action to be taken when the rule evaluates True:

NEWPASS=	Specifies the new Password or User Class number to be supplied if the Rule evaluates True. May either specify a literal password, or (preferably) a User Class number (with a '#' prefix). If specifying a literal password, the password should be quoted if it contains any non-alphanumeric or lower-case letters (unquoted strings are up-shifted). TurboIMAGE User Class numbers lie in the range 1:63. User Class number #0 means 'use null password'. User Class number #64 means "switch logon to Database creator and use the ';' creator password". For more information, refer to the **NEWPASS= Keyword Syntax** and subsequent discussions in the **Advanced Topics** section.

Use of Multiple USER= Keywords

If any keyword is followed by a list of values, the keyword evaluates True if any of the keyword values match. There is no obvious way to specify that the keyword should only evaluate True if multiple keyword values are AND'ed together.

There are situations where this may be desirable, for example if a rule needs to be limited to Jobs that log on one way, but that are also streamed by users who log on another way.

This can be achieved as shown here:

```
$DB-PASS

     USER=JOHN,@.@:i
     USER=MARY,@.@:J
     NEWPASS=#25
```

This rule takes effect for Jobs that log on as MARY,@.@, but which are also streamed by JOHN,@.@ (the ':I' suffix means 'Indirectly Streamed by').

Another example involves the need to check the name of the current program, as well as the name of the 'father' process.

The following rule evaluates True if the current program is SUPRTOOL.PUB.ROBELLE or if the father process is located in the CODE2.SGAII group:

```
$DB-PASS

     PROG=SUPRTOOL.PUB.ROBELLE
          @.CODE2.SGAII:1
     NEWPASS=#10
```

while the following rule only evaluates True if both conditions are True:

```
$DB-PASS

     PROG=SUPRTOOL.PUB.ROBELLE
     PROG=@.CODE2.SGAII:1
     NEWPASS=#10
```

$DB-PASS Rule Scanning

The CONFIG file $DB-PASS section is scanned whenever a program attempts to open a Database. Scanning proceeds by evaluating each Rule until all the Keywords in a Rule match, and the Rule evaluates True. If any Keyword specifies a list of matching values, the Keyword matches if any of the list of values matches.

Example:

```
$DB-PASS

     BASE=DB.GROUP.ACCT
          @.db.sys
     USER=JOHN,@.@
          MARY,@.@
          STEVE.APPS
     PROG=@.prog.@
     NEWPASS=#10
```

This rule is interpreted as:

```
IF  (   Base is "DB.GROUP.ACCT"
     OR Base is "@.DB.SYS"
    )

AND (   User is Job or Session "JOHN,@.@"
     OR User is Job or Session "MARY,@.@"
     OR User is Job or Session "STEVE.APPS"
    )

AND (   Prog is "@.PROG.@"
    )

THEN
     NewPass is User-Class #10
```

The rule is terminated by a blank line.

This $DB-PASS rule will replace the supplied Database password with whatever Password is currently defined for Password Class 10, for users John, Mary, or Steve while opening database DB.GROUP.ACCT, or any database in DB.SYS, while running any program in @.PROG.@. It has no effect for any other users, or when these Users run any other programs or open any other databases.

Order of General and Specific $DB-PASS Rules

Rules are scanned until a rule evaluates True. This implies that more specific rules should be specified before more general rules. For example, if the following rules are defined:

```
$DB-PASS

     # Rule #1: allow DB accessors user class 10 access:
     BASE=DB.GROUP.ACCT
     NEWPASS=#10

     # Rule #2: allow DB accessors via QUERY creator access:
     BASE=DB.GROUP.ACCT
     PROG=QUERY@.PUB.SYS
     NEWPASS=#64
```

then the second rule will never take effect, as users running QUERY.PUB.SYS will trigger Rule #1 first.

If it is intended that users running QUERY be granted Creator access, the rules should instead be specified in the following order:

```
$DB-PASS

     # Rule #1: allow DB accessors via QUERY creator access:
     BASE=DB.GROUP.ACCT
     PROG=QUERY@.PUB.SYS
     NEWPASS=#64

     # Rule #1: allow DB accessors user class 10 access:
     BASE=DB.GROUP.ACCT
     NEWPASS=#10
```

USER= Keyword Context Qualifiers

The USER= Keyword supports a very powerful syntax extension that allows a context qualifier to be appended to each User name. The qualifier allows the specified User name to be limited to Jobs or Sessions. It also allows any Job steamed by a specified User name to be qualified.

Example:

```
$PROGSET

     Plist-01 =
        @.PUB.SYS
        PR@.DBA.ADM

$DB-PASS

     PROG = ^PLIST-01
     BASE = @.pub.sys
     USER = JOHN,@.@:JS
            MARY,@.@:S
            MANAGER.SYS
            STEVE,MGR.APPS:SI
     NEWPASS = #5
```

This $DB-PASS rule applies to all programs defined in the 'PLIST-01' $PROGSET section, when they DBOPEN any database in PUB.SYS, and when the user running the program falls into any of the following cases:

- User JOHN,@.@ when logged on as a Job or Session.
- User MARY,@.@ when logged on as a Session
- User MANAGER.SYS when logged on as a Job or Session (the default context)
- User STEVE,MGR.APPS when logged on as a Session, or when the user STEVE,MGR.APPS streams a Job that logs on with ANY logon.

The ':I' context qualifier is potentially a very powerful extension to the traditional Job/Session context.

Partially-Qualified File and User Names

If any rule specifies the BASE=, CMDFILE=, FILE=, OPENFILE=, or STDIN= keywords with partially-qualified filenames, the filename will be interpreted as being padded with '.@' elements.

For example, this keyword:

```
BASE=DBA
      DBB.PUB
```

is interpreted as:

```
BASE=DBA.@.@
      DBB.PUB.@
```

Similarly, if a rule specifies the USER= keyword with partially-qualified User names, User names will be interpreted as having trailing '.@' elements.

For example, this keyword:

```
USER=MANAGER
      JOHN,@
```

is interpreted as:

```
USER=MANAGER.@
      JOHN,@.@
```

Unqualified User names are interpreted as MPE user names, and not as Job/Session names.

Pattern-Matching User Names

Sites that have implemented advanced security using Vesoft's Security/3000 product often require that online Users specify a unique job/session name (i.e. "HELLO JOHN,MANAGER.SYS") when logging on. IDent allows logons like this to be matched using an explicit User name, or using a wildcard User name, e.g. "@,MANAGER.SYS".

To simplify creating rules that match users who logon with job/session names as well as those who logon without a job/session name, IDent uses special pattern matching logic to ensure that rules that specify "@,User.Account" will also match Users who logon without specifying any job/session name.

Using CMDFILE= and UDC= with Nested Command Files and UDCs

IDent allows the CMDFILE= and UDC= keywords to be used within rules in the $LOG-USER and $DB-PASS sections of the CONFIG file. In cases where an application is run from a Command File or UDC that is itself called from within other Command Files or UDCs, the CONFIG keywords must specify the name of the last Command File and/or UDC that were executed before the application program was run.

If you need to detect the names of UDCs or Command Files that were run 'higher' in the hierarchy than the 'current' UDC or Command File, the OPENFILE= keyword may provide a solution, when used with a relative process number suffix.

The OPENFILE= Keyword

The OPENFILE= keyword may be used to limit rules to programs that have specific Files open. The keyword may also be used to match wildcards.

By default, OPENFILE checks if the current program has the specified file open. You can also append ':*n*' to the filename, where '*n*' refers to the generation number of the father process to check. ':0' refers to the current program, ':1' refers to the father of the current program, ':2' refers to the grandfather of the current program, etc.

For example, if a program was :RUN directly from the CI, then specifying "OPENFILE=file.group.acct:1" means "evaluate True if the CI has file.group.acct open".

As processing the OPENFILE= Keyword has higher overhead than other keywords, it's recommended to place this at the end of any rule. This will minimize the processing overhead by allowing other more efficient keywords to be used to filter out non-qualifying calls first.

For more information on use of this keyword, refer to **Debugging OPENFILE with Vesoft's MPEX** in the **Advanced Topics** section.

Section 8: CONFIG File $LOG-USER Section

The $LOG-USER keyword is followed by rules that define which Users are included or excluded from User Input Logging.

The CONFIG file $LOG-USER section is scanned when every session first logs on. Each rule in the file is evaluated (starting from the top) until the first rule that matches the active Job/Session is found. Once a matching rule is found, scanning of the CONFIG file $LOG-USER section stops.

There is no limit on the number of $LOG-USER rules, although you should try to keep the total number as small as possible to reduce processing overhead. $LOG-USER section rules are scanned in-order every time a Job/Session logs on, until the first rule is found that evaluates True.

This means that in general, specific rules intended to match users should be placed at the start of the $LOG-USER section, and general rules intended to match groups of users should be placed at the end of the $LOG-USER section.

A rule may reference a user by identifying the Session ID, LDev number, or IP Address that the User logs in from. Wildcards may be used with all types of rules.

Each $LOG-USER section Rule may contain one or both of the following keywords which together limit the scope of the rule:

USER=	Limits the rule to the specified User(s). May specify one or more User names, including wildcards. User names specify the MPE User and Account name, and may contain an optional leading Job/Session name (with a ',' delimiter). May refer to $USERSET section User-Set definitions by prefixing the User-Set name with a '^'. User name specifications may include a trailing ':' and any of 'J', 'S', 'I', or '@'. ':J' limits the context to Jobs, ':S' limits the context to Sessions, ':I' (for 'Indirect') limits the context to any Job streamed by the specified User. ':@' is equivalent to specifying ':JSI'; the default USER= context qualifier is ':JS'.
TERM=	Limits the rule to users logged on the specified LDev number(s) or IP address(es). May specify one or more LDev numbers and/or IP Addresses, including wildcards. Simple numbers are interpreted as LDev numbers; fields that contain embedded '.'s are interpreted as IP Addresses.

By default, $LOG-USER rules enable logging for the matching users; rules can also disable logging for matching users, using the following keyword:

[NO]LOG	LOG (the default state) causes matching users to be logged. NOLOG causes matching users to be excluded from logging.

$LOG-USER section rules may contain the following keywords which control the handling of IDent Log files when a User logs off (by default Log files are retained when Users log off):

[NO]LOG-DISCARD	LOG-DISCARD discards IDent Log files when monitored Users log off, unless IDent also logged at least one IDent "Access-allowed" or "Access-blocked" message to the Log file. By default (or if NOLOG-DISCARD is specified), IDent Log files are retained even if no Access messages were logged. NOLOG-DISCARD may be used to reverse the effect of LOG-DISCARD specified globally in the $CONFIG section.
[NO]LOG-PRIV-BOOST	LOG-PRIV-BOOST generates a Log message if the User acquires Privileged capabilities by running a 'GOD' program. When LOG-DISCARD is also specified this message will prevent the IDent Log file from being discarded. By default (or if NOLOG-PRIV-BOOST is specified), no message is logged when a 'GOD' program is run. NOLOG-PRIV-BOOST may be used to reverse the effect of LOG-PRIV-BOOST specified globally in the $CONFIG section.
[NO]LOG-PRIV-LOGON	LOG-PRIV-LOGON generates a Log message if a Privileged User logs on. When LOG-DISCARD is also specified this message will prevent the IDent Log file from being discarded. By default (or if NOLOG-PRIV-LOGON is specified), no message is logged when a Privileged logon occurs. NOLOG-PRIV-LOGON may be used to reverse the effect of LOG-PRIV-LOGON specified globally in the $CONFIG section.
[NO]LOG-STREAM	LOG-STREAM generates a Log message if the User STREAM's any job. When LOG-DISCARD is also specified this message will prevent the IDent Log file from being discarded. By default (or if NOLOG-STREAM is specified), no message is logged when a Job is Streamed. NOLOG-STREAM may be used to reverse the effect of LOG-STREAM specified globally in the $CONFIG section.

$LOG-USER section rules may optionally contain either of the following keywords which defer the start of User logging until a User acquires Privileged capabilities:

[NO]LOG-PRIV-ONLY	LOG-PRIV-ONLY only logs User input while Users are Privileged. By default (or if NOLOG-PRIV-ONLY is specified), User input is logged regardless of User capabilities. When LOG-PRIV-ONLY is specified, User Input logging starts when a User acquires Privileged capabilities, and stops if the User reverts to their original capabilities. NOLOG-PRIV-ONLY may be used to reverse the effect of LOG-PRIV-ONLY specified in the $CONFIG section.
[NO]LOG-PRIV-NOSTOP	LOG-PRIV-NOSTOP starts logging User input when User gain Privileged capabilities, and keeps logging from that point on. By default (or if NOLOG-PRIV-NOSTOP is specified), User input is logged regardless of User capabilities. When LOG-PRIV-NOSTOP is specified, User Input logging starts when a User acquires Privileged capabilities, and then continues until the User logs off. NOLOG-PRIV-NOSTOP may be used to reverse the effect of LOG-PRIV-NOSTOP specified in the $CONFIG section.

Examples:

```
$USERSET

   PM-ONLINE =
      @.@:S

$LOG-USER

   USER=@.SYS
        @.ACCT2
   LOG-PRIV-BOOST
   LOG-PRIV-LOGON

   TERM=20
        10.1.@.@
   LOG-PRIV-ONLY

   USER=JOHN,@.@:S

   USER=^PM-ONLINE
```

The CONFIG file is read until the first matching rule is found.

If user JOE,MANAGER.SYS logs on from LDev 20, and the following rules are defined:

```
$LOG-USER

    TERM=20
    NOLOG

    USER=JOE,@.@
```

then the session will not be logged.

If the following rules are defined:

```
$LOG-USER

    USER=MANAGER.SYS

    USER=JOE,MANAGER.SYS
    NOLOG
```

then the session will be logged.

By default, if no rule is found that matches a new Job or Session, then the Job or Session will not be logged. To change this to log every session by default, just add a 'USER=@.@:S' rule at the end of the $LOG-USER section.

Logging when Users have acquired Privileged Capabilities

When a $LOG-USER section rule using the LOG-PRIV-ONLY or LOG-PRIV-NOSTOP keywords evaluates True, special logic is activated that will detect when a user has acquired SM or PM capabilities, and start logging all input while the user is privileged. If the LOG-PRIV-NOSTOP keyword is specified, logging will continue if the User reverts to their original capabilities.

There are three ways that a User can trigger Logging by their capabilities:

- User has SM or PM capability permanently assigned to it.
- User runs a capability-boosting program, like Vesoft's GOD program.
- User uses a logon-switching program, like MPEX's %CHLOGON command.

When IDent detects that a User has acquired SM or PM capability, it will write a message to the Job or Session's IDent Log file that identifies the series of commands that resulted in the acquisition of boosted capabilities, and it will then start logging all User input. If the User subsequently loses SM and PM capabilities, another message will be written to the IDent Log file.

This example shows a typical trace of a User running GOD from inside another program:

```
16:27/0089/ ============================================================
16:27/0089/ IDent/3000  (c) Paul Taffel Consulting (PTC), 2001-2010
16:27/0089/
16:27/0089/ #S3291 logged on as TEST1.DEV
16:27/0089/ LDev 5; IP Address 192.168.111.5
16:27/0089/ MON, SEP  3, 2007  4:27 PM
16:27/0089/
16:27/0089/ Keywords: LOG-DISCARD, LOG-PRIV-BOOST, LOG-PRIV-LOGON.
16:27/0089/ ============================================================
16:28/0090/
16:28/0090/ User is elevating capabilities: ----------------------------
16:28/0090/ :qedit
16:28/0090/ /%god
16:28/0090/ User has acquired SM capability ----------------------------
16:28/0090/
16:28/0090/ /exit
16:28/0087/ :dbutil
16:28/0089/ >>show dbx all
16:28/0089/ >>exit
```

When the $CONFIG section OP-IS-PM keyword is specified, OP capability is also tested for. When this keyword is absent, OP users will only trigger the capability check when they also have SM or PM capability.

If the LOG-PRIV-ONLY keyword is specified, logging of a User that is triggered by capability boosting will stop if the user reverts to their original non-privileged capabilities (typically by running Vesoft's MORTAL program). If the LOG-PRIV-NOSTOP keyword is specified, logging of a User who has acquired privileged mode will continue even after they revert to their original capabilities.

Section 9: CONFIG File $DB-ACCESS Section

The $DB-ACCESS section contains rules that define which TurboIMAGE Datasets require access to be restricted, and whether access attempts should be logged.

Up to 32 different $DB-ACCESS section rules may be defined. Each rule specifies the database and dataset(s) that it applies to, the restrictions that are to be applied, and optional keywords that further limit the scope of each rule. Individual rules can apply to single databases, or can use wildcard patterns to qualify multiple databases.

$DB-ACCESS section rules can restrict read and write access to individual Datasets, and can independently specify that read or write access to individual Datasets be logged.

$DB-ACCESS rules are applied the first time that a program attempts to open any dataset; they protect against access via the TurboIMAGE intrinsics, but also against programs (like Robelle's SUPRTOOL) that can bypass TurboIMAGE and access datasets directly. Only one attempt is logged per dataset per accessing program: this is to avoid generating an avalanche of duplicate messages when programs make repeated access attempts.

Individual $DB-ACCESS File protection Rules may contain any number of the following key-words. Each specified keyword must evaluate True for the Rule to match. If any keyword evaluates False, the entire Rule will be skipped, and the next $DB-ACCESS file protection Rule will be evaluated. If no rules match, the dataset open request will be allowed.

$DB-ACCESS section rules may contain one or more of the following keywords which together limit the scope of the rule:

BASE=	Limits the rule to the specified Database(s). May specify one or more Database names, including wildcards.
CMDFILE=	Limits the rule to programs launched by the specified MPE Command File(s). May specify one or more Command File names, including wildcards.
OPENFILE=	Limits the rule to programs that have the specified File(s) open. May specify one or more File names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific file(s) open, by appending a '*:n*' generation number suffix to the file name specification, as described in the PROG= keyword description below.

PROG=	Limits the rule to the specified Program(s). May specify one or more Program names, including wildcards. May also refer to $PROGSET section Program-Sets by prefixing Program-Set names with a '^'. This keyword may be used limit the rule to programs with specific parent 'ancestor' processes by appending '*:n*', where a '*n*' is the generation number of the ancestor. ':0' refers to the current process, ':1' refers to its 'father', and ':2' refers to its 'grandfather', etc. The default generation suffix is ':0'. You cannot specify '*:n*' process generation suffixes with '^' prefixed Program-Set names (although Program-Set definitions can themselves contain generation suffixes inside the $PROGSET section).
STDIN=	Limits the rule to programs that have the specified STDIN file(s) open. May specify one or more File names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific STDIN file(s) open, by appending a '*:n*' generation number suffix to the file name specification, as described in the PROG= keyword description above.
TERM=	Limits the rule to users logged on the specified LDev number(s) or IP address(es). May specify one or more LDev numbers and/or IP Addresses, including wildcards. Simple numbers are interpreted as LDev numbers; fields that contain embedded '.'s are interpreted as IP Addresses.
UDC=	Limits the rule to programs launched by the specified MPE UDC(s). May specify one or more UDC names, including wildcards.
USER=	Limits the rule to the specified User(s). May contain one or more User names, including wildcards. User names specify the MPE User and Account name, and may contain an optional leading Job/Session name (with a ',' delimiter). May refer to $USERSET section User-Set definitions by prefixing the User-Set name with a '^'. User name specifications may include a trailing ':' and any of 'J', 'S', 'I', or '@'. ':J' limits the context to Jobs, ':S' limits the context to Sessions, ':I' (for Indirect') limits the context to any Job streamed by the specified User. ':@' is equivalent to specifying ':JSI'; the default USER= context qualifier is ':JS'.

$DB-ACCESS section rules may contain the following keyword which specifies the action to be taken when all keywords in the rule evaluate True:

SETS=	Specifies a list of Dataset numbers, and restrictions to be applied to all attempts to access those Datasets. Contains a ',' delimited list containing one or more Dataset numbers (or '@'), followed by '**;ACTION=**', followed by a ',' delimited list of one or more of the following keywords: **ALLOW .....** Allow all access (same as 'READ,WRITE'). This is the default if no ACTION is specified. **BLOCK .....** Block all access (same as 'NOREAD,NOWRITE'). **[NO]READ .....** Block or allow attempts to open Dataset(s) for read access. **[NO]WRITE .....** Block or allow attempts to open Dataset(s) for write access. **[NO]LOGALLOW ..** Log when the rule is applied and access is Allowed. **[NO]LOGBLOCK ...** Log when the rule is applied and access is Blocked. **[NO]LOG .....** Log when the rule is applied and access is Allowed or Blocked (same as specifying 'LOGALLOW,LOGBLOCK'). Multiple SETS= lines may be specified, and a given Dataset may also be referenced by more than one SETS= lines.

When a rule includes the ACCESS=LOG keyword, special logic is used to limit the number of access messages that are logged. Only the first read or write access attempt for each Dataset is logged for each run of a program. If the first attempt is a read access attempt, the first write attempt will also be logged.

The following example shows a database protected by two Rules. The first Rule defines which users and programs can access protected datasets (with logging); the second Rule causes all other access attempts to be blocked (with logging).

```
$DB-ACCESS

    # First rule defines user allowed to access Sets 1,3,11
    BASE=DBX.DATA.ACCT
    USER=@,MANAGER.SYS
         @.USERS.ACCT
    PROG=@.PUB.ACCT
    SETS=1,3,11; ACTION = ALLOW, LOG
```

```
# Second rule prevents other users accessing Sets 1,3,11
BASE=DBX.DATA.ACCT
SETS=1,11; ACTION = NOREAD, LOG
SETS=3   ; ACTION = NOWRITE, LOG
```

$DB-ACCESS section rules are checked in-order for matches when any program attempts to open any Dataset file. In the above case, if User MANAGER.SYS attempts to open Dataset #11 while running PROG1.PUB.ACCT, the first rule will match, access will be allowed, and the successful access attempt will be logged.

If User JOE.ACCTS attempts to open Dataset #3 for write access, the second rule will match, the access attempt will fail, and the unsuccessful attempt will be logged. If JOE.ACCTS attempts to access any Datasets other than #1, #3, or #11, the attempt will succeed.

Examples of the Console messages displayed in response to Users MANAGER.SYS and JOE.ACCTS attempting to access Datasets in DBX.DATA.ACCT are shown below:

```
Allowed Write access: Set 1:ACCTS of DBX.DATA.ACCT by #S134:
MANAGER.SYS using QUERY from 10.1.8.145

Blocked Write access: Set 3:HISTORY of DBX.DATA.ACCT by #S135:
JOE.ACCT using QUERY from 10.1.8.152
```

If the Dataset open attempt matches a rule that contains a wildcard BASE= specification, the resulting message will only reference the BASE= pattern, and not the specific database name.

For example, if the above example contained the following keyword:

```
BASE=DBX.@.@
```

then Console messages would display the following:

```
Allowed Write access: Set 3:HISTORY of DBX.@.@ by #S3705:
MANAGER.SYS using QUERY from 192.168.111.5
```

Section 10: CONFIG File $FILE-ACCESS Section

The $FILE-ACCESS section contains rules that define which files require special access processing to restrict file access and purge attempts, and whether these attempts should be logged.

Up to 32 different $FILE-ACCESS section rules may be defined. Each rule specifies the file or fileset that it applies to, the restrictions that are to be applied, and optional keywords that further limit the scope of each rule.

$FILE-ACCESS section rules can restrict file access in the following ways:

1) Protect Files against being Overwritten or Purged ("ACTION=PROTECT")

This is the most comprehensive protection mechanism offered by IDent. It protects specified files against being Purged by any mechanism, and protects the contents of the specified files from being overwritten or erased. It still allows files to be read from and appended to.

Protection is provided against all of the following:

- :PURGE and :PURGELINK commands
- :RENAME command
- :FILE equations that specify ;DEL disposition
- Posix shell's rm and mv commands
- :PURGEGROUP and :PURGEACCT commands
- Programmatic access to all of the above.
- :STORE command ;PURGE keyword.
- :RESTORE command is prevented from restoring on top of a protected file.
- Overwriting file contents, including resetting file EOF.

2) Protect Files against being Opened ("ACTION=BLOCK")

IDent can block attempts to open the file using any MPE mechanism. This keyword prevents files being opened for Read access (as well as Write, Append, Purge, or Run access).

This keyword is not the primary way to implement file protection, and should only be used when no alternative solutions exist.

3) Protect Files against being Run ("ACTION=NORUN")

IDent can block attempts to :RUN a program file. This will block use of the file by the :RUN and :NEWCI commands, the Posix shell's exec command, and by programmatic calls to CREATE, CREATEPROCESS and exec.

For information on the limitations of this Keyword, refer to the **Limitations on use of $FILE-ACCESS Section ACTION=NORUN Keyword** discussion in the **Advanced Topics** section.

$FILE-ACCESS section rules can also specify if failed access attempts should be logged or not. A message will be generated for all blocked access attempts if the ACTION=LOG keyword is specified, and if the $CONFIG section contains one or more of the keywords BLOCK-TO-LOG, BLOCK-TO-TERM or BLOCK-TO-CONSOLE. Messages are also generated if the access attempt is allowed, and any of the keywords ALLOW-TO-LOG, ALLOW-TO-CONSOLE, or ALLOW-TO-TERM are present. If ACTION=LOG is not specified, not indication will be given if a rule triggers and file access is allowed or blocked.

IDent $FILE-ACCESS rules are primarily intended to allow critical log files to be protected against tampering by any user, including users with SM capability. They can also be used to protect a limited number of files against being run or opened by unauthorized users, but a maximum of 32 $FILE-ACCESS rules can be active within any job or session. The total should also be limited to as small a number as possible to minimize processing overhead.

NOTE: $FILE-ACCESS section rules cannot be used to protect TurboIMAGE datasets.

TurboIMAGE datasets may be protected against unauthorized modification by defining rules in the $DB-ACCESS section.

WARNING: IDent $FILE-ACCESS rules are applied using very low-level traps that intercept all file access on the system. It is possible to cause severe problems that could make system operation impossible if rules are improperly defined. IDent will impose exactly the access restrictions that you define, and it is possible to lock all users out of the system.

There is no secret 'back-door' to IDent: if you lock yourself out the only solution is to re-boot your system without activating IDent.

When any program attempts to open any file, the list of File protection rules is scanned for the first rule that matches. If a matching rule is found, the rule's ACTION= keyword will be used to determine if the file access should be blocked or allowed

Each $FILE-ACCESS section File protection rule may contain any number of the following key-words. Each specified keyword must evaluate True for the Rule to match. If a keyword evaluates False, the entire Rule will be skipped, and the next $FILE-ACCESS file protection Rule will be evaluated. If no rules match, the file open request will be allowed.

$FILE-ACCESS section rules may contain one or more of the following keywords which together limit the scope of the rule:

FILE=	Limits the rule to the specified File name (may contain wildcards). Unlike other CONFIG rule keywords, the FILE= keyword may only specify a single filename or file pattern. Each $FILE-ACCESS section rule may also only contain one FILE= keyword. If the ':EXEC' suffix is appended to the specified file name, the scope of the rule is limited to executable files (CM or NM programs or libraries).
CMDFILE=	Limits the rule to programs launched by the specified MPE Command File(s). May specify one or more Command File names, including wildcards..
OPENFILE=	Limits the rule to programs that have the specified File(s) open. May specify one or more File names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific file(s) open, by appending a '***:n***' generation number suffix to the file name specification, as described in the PROG= keyword description below.
PROG=	Limits the rule to the specified Program(s). May specify one or more Program names, including wildcards. May also refer to $PROGSET section Program-Sets by prefixing Program-Set names with a '^'. This keyword may be used limit the rule to programs with specific parent 'ancestor' processes by appending '***:n***', where a '*n*' is the generation number of the ancestor. ':0' refers to the current process, ':1' refers to its 'father', and ':2' refers to its 'grandfather', etc. The default generation suffix is ':0'. You cannot specify ':*n*' process generation suffixes with '^' prefixed Program-Set names (although Program-Set definitions can themselves contain generation suffixes inside the $PROGSET section).
STDIN=	Limits the rule to programs that have the specified STDIN file(s) open. May specify one or more File names, including wildcards. This keyword may be used to limit the rule to programs whose ancestor processes have specific STDIN file(s) open, by appending a '***:n***' generation number suffix to the file name specification, as described in the PROG= keyword description above.
TERM=	Limits the rule to users logged on the specified LDev number or IP address. May specify one or more LDev numbers and/or IP Addresses, with wildcards. Simple numbers are interpreted as LDev numbers; fields that contain embedded '.'s are interpreted as IP Addresses.

UDC=	Limits the rule to programs launched by the specified MPE UDC(s). May specify one or more UDC names, including wildcards.
USER=	Limits the rule to the specified User(s). May contain one or more User names, including wildcards. User names specify the MPE User and Account name, and may contain an optional leading Job/Session name (with a ',' delimiter). May refer to $USERSET section User-Set definitions by prefixing the User-Set name with a '^'. User name specifications may include a trailing ':' and any of 'J', 'S', 'I', or '@'. ':J' limits the context to Jobs, ':S' limits the context to Sessions, ':I' (for 'Indirect') limits the context to any Job streamed by the specified User. ':@' is equivalent to specifying ':JSI'; the default USER= context qualifier is ':JS'.

$FILE-ACCESS section rules may contain the following keyword which specifies the action to be taken when the rule evaluates True:

ACTION=	Specifies the effect(s) of the rule. Contains a ',' delimited list of one or more of the following keywords: **ALLOW .....** Allow all file access (same as "READ,WRITE,APPEND, PURGE,RUN"). This is the default if no ACTION is specified. **BLOCK .....** Block all file access (same as "NOREAD,NOWRITE, NOAPPEND,NOPURGE,NORUN"). **PROTECT .....** Protect the file (same as "NOWRITE,NOPURGE"). **[NO]READ .....** Block or Allow Read access. **[NO]WRITE .....** Block or Allow Write access. Also controls destructive-Read access to MSG files. **[NO]APPEND .....** Block or Allow Append access. **[NO]PURGE .....** Block or Allow Purge access. **[NO]RUN .....** Block or Allow Execute access. **[NO]LOGALLOW ..** Log when the rule is applied and access is Allowed. **[NO]LOGBLOCK ...** Log when the rule is applied and access is Blocked. **[NO]LOG .....** Log when the rule is applied and access is Allowed or Blocked (same as specifying 'LOGALLOW,LOGBLOCK').

Section 11: CONFIG File $CHECKSUM Section

The $CHECKSUM section contains rules that define which files should be scanned by default when the CHECKSUM program is run. The CHECKSUM program uses a cryptographically-secure checksum algorithm to detect file changes, and is designed to be run periodically. For more information on use of the CHECKSUM program, see **The IDent CHECKSUM Program** section.

Any number of $CHECKSUM section rules may be defined. Each rule specifies a single file or fileset to must be scanned; files may be specified in MPE or HFS (Posix) syntax. Each rule (including the last) must be followed by at least one blank line,

Examples:

```
$CHECKSUM

   FILE=@.PUB.SYS

   FILE=@.PUB.PTC

   FILE=/ACCT/CODE/[C-D]@
```

which leads to the following processing when the CHECKSUM program is run:

```
 :RUN CHECKSUM.PUB.PTC;PARM=1

 IDent/3000 Checksum Utility (c) Paul Taffel Consulting (PTC) 2009-2010

 IDent/3000 CHECKSUM scanning CONFIG file entries (mode 1)

 Scanning Fileset @.PUB.SYS
 Scanning Fileset @.PUB.PTC
   File changed  : OFF.PUB.PTC
   File changed  : ON.PUB.PTC
 Scanning Fileset /PTC/CODE/[C-D]@
```

Section 12: IDent Log File Handling

This section contains information on the options available for handling IDent Log Files.

Active IDent Log Files

All user input from any logged job or session is written in real-time to a single log file, saved with the name 'S#' or 'J#', in the LOG.PTC group. Job and Session logging to this file stops when the job or session logs off.

Active Log files may be viewed using the following MPEX command (even if the Job or Session being logged is still logged on):

```
%PRINT S###.LOG.PTC; WAIT=-1
```

The MPEX %PRINT command's ;WAIT=-1 keyword will update the display every second (the '-' prefix means "don't beep when updating display"). The MPE :PRINT command may also be used to view 'snapshots' of active log files.

If job or session numbers wrap-around, any pre-existing log files in the LOG.PTC group having the same file name as a new logged job or session will automatically be purged when the new user logs on.

By default, access to IDent log files in the LOG.PTC group is restricted to the user MGR.PTC, and to users with SM capability. We recommend defining a $FILE-ACCESS section file protection rule to further limit access to users logged on to a physically secure LDev or IP Address.

For more information on protecting Log Files, refer to the **Protecting IDent Files** discussion in the **Advanced Topics** section.

Archived IDent Log Files

Individual IDent Log Files are built in the LOG.PTC group, and remain in this location during the lifetime of a Job or Session. When a Job or Session logs off, the Log File is renamed into the ARCHIVE.PTC group. If the CONFIG file LOG-ARCHIVE keyword has been defined, the Log File name will be renamed according to the specified pattern.

Finally, if the CONFIG file FTP-TARGET keyword has been defined, the Log File will be FTP'd to the final destination, and the destination File name will be changed according to the specified pattern.

NOTE: IDent Log Files are left on the HP3000 in the ARCHIVE.PTC Group, and must periodically be removed to prevent this group filling up (there is a limit on the number of files that can be stored inside a single MPE Group; exceeding this limit can cause serious problems).

$CONFIG Section LOG-ARCHIVE keyword

The LOG-ARCHIVE keyword allows IDent Log files to be assigned a name that includes date and/or time-specific components when the files are renamed from the LOG.PTC to the ARCHIVE.PTC group. Incorporating elements of the current date and/or time into Log file names allows multiple files with the same Job or Session number to be archived in the same group without name collisions.

The new Log file name is specified using the CONFIG file's LOG-ARCHIVE keyword in the $CONFIG section. The syntax for LOG-ARCHIVE is:

```
$CONFIG

    LOG-ARCHIVE="filename-pattern"
```

The "*filename-pattern*" string may contain any of the following reserved identifiers – each identifier will be replaced with the corresponding current Date/Time values:

@	Job/Session Number (e.g. "S123"). Must be specified once.
Z@	Like '@' but includes leading zeros (e.g. "S00123"), a total of 6-characters.
YYYY	4-digit **Y**ear.
YY	2-digit **Y**ear.
MM	2-digit **M**onth.
DD	2-digit **D**ay-of-month.
DDD	3-digit **D**ay-of-year.
HH	2-digit **H**our (0..23).
NN	2-digit mi**N**ute (0..59).
SS	2-digit **S**econd (0..59).
G	1-digit **G**eneration (0..9 or A..Z).

The "*filename-pattern*" string may also contain other characters, as long as the combined total length of the expanded string is no longer than 16 characters (the longest filename that may be created directly inside an MPE Group), and the filename is still a legal MPE or Posix-syntax name.

The above Date/Time identifiers must be specified in upper-case to be recognized. If LOG-ARCHIVE contains any other data, the literal text will appear in the target filename.

For example, the following specification:

```
LOG-ARCHIVE="@_DDD"
```

would create the following filename for #S3851 on day 230 of the year:

```
S3851_230.ARCHIVE.PTC
```

The Generation ("G") format character allows multiple files with otherwise identical names to coexist in the ARCHIVE.PTC group. The "G" character is normally "0", but will cycle through the characters "1" through "9" and "A" through "Z" if the new file name already exists.

Using the following LOG-ARCHIVE specification will ensure that all Log file names remain legal MPE-syntax names, and can therefore be viewed using the :LISTF command:

```
LOG-ARCHIVE="@DDDG"
```

Using the following LOG-ARCHIVE specification takes advantage of the longer filenames supported by Posix. Creating Posix-syntax file names means that the contents of the ARCHIVE.PTC group must be viewed using the :LISTFILE command with Posix-syntax:

```
LOG-ARCHIVE="@-YYYYMMDDG"
```

NOTE: Don't specify a LOG-ARCHIVE string that would result in attempts to create file names longer than 16 characters – the limit for file name length for files located directly inside an MPE Group.

As the largest Job/Session number ("S16383") is six characters long, the LOG-ARCHIVE specification should not exceed 11 characters, including the "@".

FTP Configuration

IDent can be configured to automatically FTP all log files to a remote system as soon as each log file becomes available.

The actual FTP transfer is performed by a background job that IDent is creates when the program ON.PUB.PTC is run. Several CONFIG file keywords must be configured before the FTP job will run. The following $CONFIG section keywords allow the remote FTP logon to be specified:

FTP-ADDR= ***"address"***	Specifies the remote FTP System Name or IP Address.
FTP-USER= ***"user-name"***	Specifies the remote FTP User logon. *user-name* is not automatically up-shifted.
FTP-PASS= ***"password"***	Specifies the remote FTP User password. *password* is not automatically up-shifted.
FTP-PROG= ***"program"***	Specifies the location of the local FTP client program to be run. Defaults to "FTP.ARPA.SYS".
FTP-TARGET= ***"pattern"***	Specifies the remote filename. Defaults to "@". For more details, see **FTP Configuration** in the **IDent Log File Handling** section.
FTP-CMD1= ***"command"***	Specifies the first FTP command sent after logon (optional).
FTP-CMD2= ***"command"***	Specifies the second FTP command sent after logon (optional).
FTP-JOB= ***"job-file"***	Specifies name of file containing FTP background Job (optional).
FTP-START	Specifies that the FTPMGR Job is automatically started or stopped whenever IDent is enabled or disabled.
FTP-TRACE	Displays Console message when a Log file is exported via FTP.

The FTP-TARGET keyword allows the filename on the remote system to include the date and/or time that the log file was closed, as well as user-specified text. If the FTP-TARGET specification includes any of the following tokens, they will be replaced by the current date or time field:

@	Job/Session Number (e.g. "S123"). '@' or 'Z@' must be specified once.
Z@	Like '@' but includes leading zeros (e.g. "S00123"), a total of 6-characters.
YYYY	4-digit **Y**ear.
YY	2-digit **Y**ear.
MM	2-digit **M**onth.
DD	2-digit **D**ay-of-month.
DDD	3-digit **D**ay-of-year.
HH	2-digit **H**our (0..23).
NN	2-digit mi**N**ute (0..59).
SS	2-digit **S**econd (0..59).

These date/time tokens are case-sensitive: they must be specified in upper-case to be recognized.

If the FTP-TARGET specifies any other text, the literal text will appear in the target filename.

For example, the following specification

```
FTP-TARGET="YYYY-MM-DD-@.txt"
```

will create the following filename:

```
2007-10-05-S3851.txt
```

Customizing the FTP Job File

The background job that IDent uses to FTP log files to a remote system is normally generated dynamically – the actual job stream is not saved to disc as a permanent file.

If required, it is possible to modify the contents of the job stream that IDent uses to FTP log files (for example, if it is necessary to add parameters to the JOB statement). To accomplish this, use the following sample job as the template, make any required changes, save the modified file as a job stream in PUB.PTC (save the file unnumbered), and finally use the FTP-JOB keyword (described above) to tell IDent the name of the file that should be streamed. The modified job will then be used whenever IDent starts up the background FTP job.

```
!JOB FTPMGR,MGR.PTC,PUB;HIPRI
!# Job generated automatically by Ident
!SETVAR HPAUTOCONT TRUE
!SETDUMP
!CHDIR
!FILE FTP=FTP.PUB;EXC
!SETVAR FTPSTOP FALSE
!WHILE NOT FTPSTOP
!  RUN *FTP
!  IF FTPSTOP
!    TELLOP IDent FTPMGR Job Stopping
!  ELSEIF CIERROR=625
!    SETVAR FTPSTOP TRUE
!  ELSE
!    PAUSE 60
!  ENDIF
!ENDWHILE
!EOJ
```

Section 13: Logging Options

IDent can be configured to log successful and unsuccessful attempts by users to access restricted resources - these log messages may be sent to several different destinations.

This Section describes the different destinations supported by IDent, and the $CONFIG section options that configure them.

Log Message Destinations

IDent Log messages may be sent to any (or all) of five different destinations:

- System Console

  Console messages identify the name and job/session number of the user who generated the message, their IP Address or LDev number, and (for Jobs) the name and job/session number of the user who streamed the Job. The active program name is also displayed, when appropriate.

  Examples:

```
5:34/#S2749/87/IDent/3000 Disabled
5:34/#S2749/87/by #S279:PAUL,MGR.DEV on 192.168.111.5

5:35/#J5633/88/IDent/3000 already Disabled
5:35/#J5633/88/by #J563:JIDENT,MANAGER.SYS streamed by #S2750:PAUL,MGR.DEV

5:35/#J5633/88/IDent/3000 Enabled
5:35/#J5633/88/by #S285:JOE,MANAGER.SYS on LDev 20
```

- IDent Log File

  Writes single-line messages to the Job or Session IDent Log file.

  "Access-blocked" messages are identified with trailing '+'s, while "Access-allowed" and other information messages are identified with trailing '-'s. These tags allow IDent messages to be distinguished from normal logged User input.

- IDent Security Monitor Session

  Displays messages on the Terminal of any session logged on as MONITOR.PTC (the session may also be logged on with any desired Session name). This is a special user logon that IDent uses in cases where users wish to display all IDent messages on a reserved terminal screen.

The message format is similar to that sent to the System Console, although the current Day is displayed, and the process pin number is omitted. Messages will be sent to all sessions logged on as MONITOR.PTC. If no IDent Monitor session is logged on, these messages are discarded.

- User Terminal

  Displays single-line messages on the Terminal of the User who generated the message. Note that these messages may not be legible or formatted cleanly if the user is using a block-mode program.

- User Terminal Function Key Area

  Displays single-line messages over the Function Key Labels of the Terminal of the User who generated the message. These messages remain visible until the User hits <Return>.

Configuring Log Message Destinations

Messages generated fall into two categories – messages that identify actions that have been Blocked, and messages that identify actions that have been Allowed. The destination that both classes of messages are sent to may be independently configured using the following $CONFIG section keywords:

BLOCK-TO-CONSOLE	Send "Access-blocked" messages to the System Console.
BLOCK-TO-LOG	Write "Access-blocked" messages to Job/Session IDent Log file.
BLOCK-TO-MONITOR	Send "Access-blocked" messages to the IDent Security Monitor session.
BLOCK-TO-TERM	Send "Access-blocked" messages to the User Terminal.
BLOCK-TO-TERM-FKEYS	Writes "Access-blocked" messages over User Terminal function keys.

ALLOW-TO-CONSOLE	Send "Access-allowed" messages to the System Console.
ALLOW-TO-LOG	Write "Access-allowed" messages to Job/Session IDent Log file.
ALLOW-TO-MONITOR	Send "Access-allowed" messages to the IDent Security Monitor session.
ALLOW-TO-TERM	Send "Access-allowed" messages to the User Terminal.
ALLOW-TO-TERM-FKEYS	Writes "Access-allowed" messages over User Terminal function keys.

If no "BLOCK-TO-XXX" keywords are specified, "Access-blocked" messages will be sent to the System Console.

If no "ALLOW-TO-XXX" keywords are specified, "Access-allowed" messages are discarded.

"Access-Allowed" and "Access-Blocked" Log Message Format

The exact format of (and information contained within) Access-allowed and Access-blocked messages varies according to where the message is sent. There are four possible destinations: the System Console, the User's Terminal, the Job/Session IDent Log file, and the IDent Monitor Session.

- System Console Messages. These contain the following information:

- The operation that was allowed or blocked.
- The file name (or dataset name, when appropriate).
- The user name and job/session number.
- The program that was used.
- The user's location (LDev number or IP address).
- (for jobs) the User that streamed the job.

Example:

```
Blocked Write access: Set 3:USERS of DBX.PUB.APPS by #S3705:
PAUL,MGR.DEV using DBUTIL from 192.168.111.5
```

- IDent Monitor Session Messages.

The format is identical to messages sent to the System Console, with a slightly different prefix format: The current Day is displayed, and the originating process pin number is not shown.

- User Terminal Messages. These contain the following information:

- The operation that was allowed or blocked.
- The file name (or dataset name & number, when appropriate).

Example:

```
Allowed Read access: Set 2:CCARDS of DBX.PUB.APPS
```

- IDent Log File Messages. These contain the following information in a single line:

  - The operation that was allowed or blocked.
  - The file name (and dataset name & number, when appropriate).

  All "Access-blocked" and "Access-allowed" messages written to an IDent Log File are padded with trailing '+' or '-' characters, to distinguish the messages from regular user input logged to the same file.

  Example:

```
Blocked Write access: Set 3:USERS of DBX.PUB.APPS ++++++++++++++++++
Allowed Read access: Set 2:CCARDS of DBX.PUB.APPS ----------------
```

  If the $CONFIG section LOG-VIA keyword is also specified, all messages sent to any IDent Log File will be followed by a second line which shows the series of all programs, UDCs, and Command File names executed in the access attempt.

  Example:

```
Blocked Write access: Set 3:USERS of DBX.PUB.APPS ++++++++++++++++++
 via DBUTIL via :DBU ++++++++++++++++++++++++++++++++++++++++++++++
```

Logging Reflection File Transfers

Although IDent can log all user input on the system, by default (and by design) it does not log file transfer operations performed using the Reflection terminal emulator. If attempts are made to transfer files that are covered by IDent $FILE-ACCESS rules, the file open attempts will be logged, but no log entry will normally be made if transferred files are not subject to file access control rules.

The CONFIG file LOG-PCLINK keyword can specified – it causes all file transfers initiated by Reflection to be logged, regardless of whether the files are covered by other $FILE-ACCESS rules.

Internally, Reflection uses a file transfer program normally stored as PCLINK.PUB.SYS to perform file transfers. As many sites maintain several versions of this program, IDent monitors all file opens performed by any program that matches the fileset PCLINK@.PUB.SYS. IDent generates a log entry when these programs open files.

This log file fragment demonstrates how a file transferred using Reflection is logged, when the LOG-VIA keyword is also specified. The log file also logs the status of the transfer: 'F' for failure, 'S' for success.

```
16:48/0083/ :qed
16:49/0066/ RUN PCLINK2.PUB.SYS
16:49/0081/ Allowed Read access to JIDEN505.JFX.DEV ------------------
16:49/0081/  via PCLINK2.PUB.SYS via QEDIT via :QED ------------------
16:49/0066/ S
```

The IDent Security Monitor Session

IDent supports writing copies of all Log Messages to a terminal other than the System Console. This feature allows a terminal to be dedicated to displaying IDent Log Messages as they are generated, and can simplify detection of security breaches.

To enable logging to the IDent Monitor session, specify either ALLOW-TO-MONITOR and/or BLOCK-TO-MONITOR keywords in the CONFIG file $CONFIG section, and log on as MONITOR.PTC on the terminal(s) designated to receive copies of all IDent security messages.

By default the IDent Monitor session must be logged on as MONITOR.PTC (any session name may also be specified); it is possible to configure a different user logon for the Monitor by using the MONITOR-ACCT keyword in the CONFIG file $CONFIG section to specify another account name (the 'MONITOR' user-name part of the logon is hard-coded and cannot be changed).

IDent will write Log messages to any number of sessions logged on as MONITOR.PTC, although the number should be kept as low as possible for performance reasons.

The MONITOR.PTC user is created automatically when IDent is installed. The user is set with minimum capabilities, and with its home group set to the MONITOR.PTC group.

When the STATUS-TO-MONITOR keyword is specified in the CONFIG file $CONFIG section, IDent will also write status messages to the Monitor session when any of the programs ON, OFF, or CHECKSUM in PUB.PTC are run.

MONITOR messages identify the originating user's logon ID, in addition to the message text.

Section 14: The IDent CHECKSUM Program

IDent provides the ability to track critical filesets for changes by periodically calculating and storing cryptographic checksums. Cryptographic checksumming algorithms calculate message digests (or 'signatures') as a function of the entire contents of a file. There is a very high probability that any change whatsoever to the contents of a file will result in a different checksum. It is essentially impossible to modify a file in any way without the checksum also changing.

The IDent CHECKSUM program uses the Whirlpool[1] cryptographic hash function to calculate 512-bit checksums for any desired file. These checksums can be displayed (formatted in either base-64 or hexadecimal notation), or stored in an internal index file. You can perform a Checksum scan as often as desired; the CHECKSUM program can detect if file checksums have changed since the previous scan was performed, and can also infer when files have been created or deleted.

The CHECKSUM program is located in the PUB.PTC group. The program may be run in several ways: by default it scans all files that have been specified inside the $CHECKSUM section of the CONFIG file; you can also scan any file or fileset at run-time, using the :RUN command ;INFO string. The CHECKSUM program may only be run by users with SM or OP capability.

The CHECKSUM program provides several modes of operation. By default, the following menu is displayed when CHECKSUM is run:

```
IDent/3000 Checksum Utility (c) Paul Taffel Consulting (PTC) 2009-2010

Checksum run with INFO="@.@.SYS"

Available options:

   1 .. Identify added, changed files.
   2 .. Identify added, changed, deleted files.
   3 .. Identify added, changed, deleted, unchanged files.

 100 .. Display qualifying file names

 101 .. Recalculate & Display file checksums (hex encoded)
 102 .. Recalculate & Display file checksums (base64 encoded)

 201 .. Display saved checksums (hex encoded)
 202 .. Display saved checksums (base64 encoded)

Select option:
```

[1]For information on cryptographic checksum algorithms, and on Whirlpool in particular, see:

http://en.wikipedia.org/wiki/Cryptographic_hash_function
http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html

You can also use the :RUN command ;PARM keyword to specify an Option Number, in which case this menu is not displayed.

CHECKSUM Menu Options

The various CHECKSUM Menu options are summarized below:

1	Identify added, changed files. Recalculates the checksum for every qualifying file, compares it to the stored checksum, and identifies new or changed files. Saves checksum in Index file.
2	Identify added, changed, deleted files. Recalculates the checksum for every qualifying file, compares it to the stored checksum, and identifies new, changed and deleted files. Saves checksum in Index file.
3	Identify added, changed, deleted, unchanged files. Recalculates the checksum for every qualifying file, compares it to the stored checksum, and identifies new, changes, deleted, and unchanged files. Saves checksum in Index file.
100	Display qualifying file names. Shows the list of filenames that would be processed, without performing any checksumming or checking.
101	Recalculate & Display file checksums (hex encoded). Recalculates and displays the checksum for every qualifying file; does not save the checksum, and does not verify if files are new or changed. Checksums are formatted as 8 groups of 16 hex characters: 128 bytes total.
102	Recalculate & Display file checksums (base64 encoded). Recalculates and displays the checksum for every qualifying file; does not save the checksum, and does not verify if files are new or changed. Checksums are formatted as an 88 byte base64 encoded string.
201	Display saved checksums (hex encoded). Displays the saved checksum for every qualifying file; does not verify if files are new or changed. Checksums are formatted as 8 groups of 16 hex characters: 128 bytes total.
202	Display saved checksums (base64 encoded). Displays the saved checksum for every qualifying file; does not verify if files are new or changed. Checksums are formatted as an 88 byte base64 encoded string.

CHECKSUM Operation Notes

The following notes apply to aspects of CHECKSUM operation:

Configuring which File(s) to scan

You can configure which files and/or filesets should be scanned by the CHECKSUM utility by making entries in the CONFIG file's $CHECKSUM section; you can also run the utility and specify a file or fileset at runtime, using the ;INFO string to specify a single filename or fileset. CHECKSUM accepts filenames in MPE or HFS (Posix) format; it will automatically include HFS-syntax filenames even if a fileset if specified using MPE format. CHECKSUM displays a menu when run without the ;PARM keyword specified; this menu identifies whether the utility will scan a user-supplied fileset, or whether it will scan all CONFIG file $CHECKSUM section filesets.

The Checksum Index File

When Menu Options 1 through 3 are specified, IDent stores each calculate checksum in an Index file. The index file is a privileged KSAMXL file, stored as CHECKSUM.LOG.PTC.

IDent compares newly-calculated checksums with the last checksum stored in the Index file, and uses the result to determine if a file is new, changed, or deleted since the last scan. The Index file stores a single checksum for each indexed file. The Index file can be backed-up for extra security, and can be restored at any time if required.

Scanning Message and FIFO Files

Checksum can safely be used to scan Message and FIFO files – it does not read records destructively from these files, and will not block on empty files.

Limitations on Scanning Files

CHECKSUM is able to scan all file types, with the exception of directories and certain privileged files in PUB.SYS (such as HPGID). These files can't be opened by other programs either, and are therefore not vulnerable to tampering.

If CHECKSUM is unable to open any file, it will not display an error indication if the file is qualified as part of a wildcard fileset specification; it will, however, display an open error if the filename is specified explicitly. CHECKSUM will not detect if directory files are added, changed, or deleted, although it will detect changes to all files inside a directory.

CHECKSUM will scan files with Posix-syntax file names, but it will only track checksums for files whose path is 64 bytes long or shorter. Files with longer paths will only be processed when using options 101 through 103.

Special File Access Techniques

Special logic is used to ensure that running Checksum does not alter file last-access timestamps; Checksum also bypasses any file access restrictions defined using IDent's $FILE-ACCESS section rules.

Performance Considerations

The 'Whirlpool' checksum algorithm used by CHECKSUM is quite computationally intensive. The program must read every single record in a file to derive the checksum, hence the program can run for lengthy periods on slow systems. It is recommended to limit its use initially to files that are critical to MPE operation, such as the @.@.SYS fileset. If you have a fast enough machine available, it may be feasible to expand the use of CHECKSUM to include user filesets too.

Checksum Calculation

Currently file checksums are derived from every data record (including user labels) in the file; the calculation includes record-length information, and hence any change to the contents of a checksummed file will result in a change to the file's checksum (the actual chance that a change to a file would result in the same checksum being generated is stated as $1\text{-}10^{-154}$). Checksums do not include file label information (file owner, Posix groupid, security settings, etc), so changes to these file attributes will not alter a file's checksum.

Special handling for CM Programs and Libraries

CM programs and libraries (PROG and SL filecodes) contain linkage tables that can be modified when the MPE/iX CM Loader process loads the file. This would result in changes to file checksums as a result of merely running the file or loading through the library. CHECKSUM uses special logic to exclude CM linkage tables from its checksum calculation, resulting in reliable and durable checksums.

Special handling for NL.PUB.SYS

The NM system library NL.PUB.SYS also contains regions that are mapped by MPE/iX, and which dynamically change. CHECKSUM attempts to exclude these dynamic areas from its checksum calculation. Note that this special logic only triggers for NL.PUB.SYS. If you copy this file to a different location, the two checksums will differ, as CHECKSUM's special logic only triggers when it scans NL.PUB.SYS.

Interrupting Operation using <Control-Y>

When run with option 3, CHECKSUM displays the name of each file after it's scanned. When run with other options, if <Control-Y> is hit while the utility is running, it will pause, display the name of the file currently being scanned, and give you the option of continuing the

scan, or aborting the run. Enter 'C' to continue, or 'A' to abort. If CHECKSUM is aborted, the system JCW is set to indicate that the program was aborted per user request.

System Variables

CHECKSUM sets three system variables when it's run; these variables track the total number of files that are reported as having been added, changed, or deleted since the last run:

CHECKSUM_ADD
CHECKSUM_CHANGE
CHECKSUM_DELETE

Deleted File Detection

Checksum entries for Deleted files are removed from the Index file when the file is reported as deleted. Subsequent runs of CHECKSUM will not find the matching Index entry, hence deleted files will only be reported once.

IDent Security Monitor and System Console Messages

IDent logs all attempts to run Checksum in modes 1 through 3 to the System Console, and to the IDent Security Monitor (when configured). The logged message identifies the fileset being scanned and the user performing the scan.

Multiple Concurrent use of CHECKSUM

It is possible to run multiple copies of CHECKSUM concurrently; in some cases (large multiprocessor systems) this may provide performance improvements.

Examples:

```
:RUN CHECKSUM.PUB.PTC;PARM=1

IDent/3000 Checksum Utility (c) Paul Taffel Consulting (PTC) 2009-2010

IDent/3000 CHECKSUM scanning CONFIG file entries (mode 1)

Scanning Fileset @.PUB.SYS
Scanning Fileset @.PUB.PTC
  File changed  : OFF.PUB.PTC
  File changed  : ON.PUB.PTC
Scanning Fileset /PTC/CODE/[C-D]@
```

Section 15: Advanced Topics

This section contains more detail on details of IDent implementation and usage.

Special Protection for Enabling & Disabling IDent

As disabling IDent turns off all of the active security protection that it provides, the following notes apply to use of the ON and OFF programs that control IDent:

- Neither the ON.PUB.PTC and OFF.PUB.PTC programs will work if they are renamed or copied to different names or locations.

- A System Console message is written when either program is run, identifying the user who ran the program, and where they are logged on.

- Users running the ON.PUB.PTC program must have SM or OP capability, with no exceptions.

- Users running the OFF.PUB.PTC program must have SM capability, with no exceptions.

- A $FILE-ACCESS section rule can be defined that further limits access to the OFF.PUB.PTC program. For example access can be limited to particular users logged on at a specific LDev number and/or IP Address. Access to the CONFIG file should also be limited using a separate $FILE-ACCESS rule, to ensure that rules cannot be altered.

See the NORUN keyword in the $FILE-ACCESS section for more details.

Rebuilding a Local CONFIG File

When any Job or Session first logs on, a privileged temporary file is created that contains a copy of all CONFIG file rules that apply to the logon. This file is typically much smaller than the 'master' CONFIG.PUB.PTC file, and speeds up rule lookup. As the file is privileged (for security reasons), it's not possible to access it using MPE commands.

If the 'master' CONFIG.PUB.PTC is modified after a Job or Session logs on, any changes will not normally take effect until the next time the Job or Session logs on. This can be a problem when setting up and testing CONFIG rules.

It is possible to force the local CONFIG file to be rebuilt from the current CONFIG.PUB.PTC, by issuing this command from within any Job or Session:

```
:RUN RESET.PUB.PTC
```

This will cause the local CONFIG file to be purged and rebuilt.

If the :CHGROUP command (or the MPEX %CHLOGON command) is used within any Job or Session, the local CONFIG file will be rebuilt as a straight copy of the 'master' CONFIG.PUB.PTC file.

This local CONFIG file contains all rules in the master file, and does not optimize rule access. This is necessary to accommodate the possibility of multiple processes each running in different logon environments.

Restricting Access to :DEBUG

The System Debug Facility can be entered by a variety of methods: the :DEBUG command can be used from the CI prompt , and the DEBUG and HPDEBUG intrinsics can be called from CM or NM code. The DEBUG intrinsic drops online users into Debug, from which they can type commands; the HPDEBUG intrinsic does the same, but can also be used by programs to execute one or more Debug commands without user intervention. The :SETDUMP command can also be used to specify Debug commands to be executed when a program aborts;

Unrestricted access to the System Debug Facility by a user or program with PM capability can be used to bypass normal security protection mechanisms, in addition IDent is unable to log commands input interactively from inside Debug. For these reasons IDent provides the ability to fully or partially restrict access to Debug, and also provides the ability to log which commands are executed programmatically using HPDEBUG.

By default, access to Debug is allowed. The following CONFIG file $CONFIG section keywords can be used to restrict access to Debug:

DEBUG-BLOCK	Blocks any attempt to enter the System Debug Facility. Equivalent to specifying both DEBUG-BLOCK-ONLINE and DEBUG-BLOCK-PROGRAM keywords.
DEBUG-BLOCK-ONLINE	Blocks attempts to enter Debug in interactive mode.
DEBUG-BLOCK-PROGRAM	Blocks attempts to enter Debug to execute predefined command sequences programmatically.
DEBUG-ALLOW-SM	Allows SM users to override the DEBUG-BLOCK-ONLINE keyword.
DEBUG-LOG	Logs attempts to enter DEBUG; these attempts cause Log files to be retained if LOG-DISCARD is also specified.
LOG-DEBUG-COMMAND	Logs the actual command(s) passed to Debug for execution.

Some HP and third-party programs may require programmatic access to Debug to function correctly, although the number of programs with this requirement is quite small. We recommend blocking both online and program access to Debug, unless you have a particular need to relax these limitations. The DEBUG-LOG-COMMAND keyword will log the exact command(s) that are executed programmatically; this should provide evidence of attempts to bypass security restrictions.

The following CONFIG file $CONFIG section keywords should impose sufficient security in most circumstances:

```
$CONFIG

   DEBUG-BLOCK-ONLINE

   LOG-DEBUG-COMMAND
```

Although IDent is unable to log interactive user input in Debug, IDent's File Access protection still functions when inside Debug.

When the DEBUG-BLOCK-ONLINE keyword prevents an attempt to use the :DEBUG command, the attempt will result in CIERR 10 being reported: "Execution of this command has been disabled".

The :SETDUMP mechanism is also disabled to stop programs dropping into Debug if they abort.

NOTE: CM programs that are blocked when attempting to call DEBUG will be terminated. This is due to technical considerations that prevent the program from continuing to execute after an attempt to call DEBUG has been blocked. NM programs can continue to execute after attempts to call DEBUG have been blocked.

Restricting Access to PEUTIL

Internally, IDent uses MPE's AIF:PE (Procedure Exits) Facility to allow it to implement many of its advanced features. HP supplies a program (PEUTIL.PUB.SYS) that allows the AIF:PE Facility to be globally disabled. As this would allow users to bypass the security features implemented by IDent, we provide the ability to control access to the PEUTIL program. When the PEUTIL-BLOCK keyword is specified, although PEUTIL may still be run, any attempt to use the PEUTIL 'disallow' command will fail.

The PEUTIL program will remain protected even if it is renamed or copied to another name or location.

Restricting Access to :PURGEGROUP and :PURGEACCT

IDent has the ability to prevent use of the :PURGEGROUP and :PURGEACCT commands to remove MPE Groups that contain files which are protected by $FILE-ACCESS rules. By default IDent does not limit use of these commands; protection is only implemented when the $CONFIG section PURGEX-BLOCK keyword is specified.

The PURGEX restriction logic is sophisticated; it will only prevent the use of a :PURGEGROUP or :PURGEACCT command if all of the following are True:

- The :PURGEGROUP or :PURGEACCT command would result in the removal of at least one file that is potentially covered by any $FILE-ACCESS section rule.

- The $FILE-ACCESS rule specifies the PROTECT keyword.

- The $FILE-ACCESS rule protect files that currently exist.

The last restriction is implemented to prevent $FILE-ACCESS rules that specify filesets like "FILE=LOG@.@.@" from preventing any :PURGEGROUP from completing. In this case, :PURGEGROUP would only be blocked when attempting to remove a Group that currently contains "LOG@" files.

The PURGEACCT restriction is all-or-nothing: a :PURGEACCT command will be blocked in its entirety if any of the Groups that are qualified contain protected files.

A Simple Mechanism to Selectively Block :PURGEGROUP

As the $CONFIG section PURGEX-BLOCK restriction only applies to Groups that contain protected files, a single $FILE-ACCESS protection rule can be defined to protect multiple MPE Groups from being purged.

If the following rule is defined:

```
$FILE-ACCESS

   FILE=PROTECT.@.@
   ACTION=PROTECT,LOG
```

then any group that contains a file called PROTECT will be protected against removal.

Restricting Access to FSCHECK

The HP-supplied program FSCHECK.MPEXL.TELESUP may be used to Purge files using low-level mechanisms which bypass the documented file system calls that IDent intercepts.

When the FSCHECK-BLOCK keyword is specified, any attempt to run FSCHECK will be blocked. By default, IDent does not block access to FSCHECK.

Internally, the FSCHECK-BLOCK keyword blocks access to the message catalog file used internally by FSCHECK. When access to the message catalog is unavailable, FSCHECK will not run. This method of blocking execution of FSCHECK program works even if the program file is renamed or copied to another name or location.

Advanced FSCHECK Protection

The $CONFIG section FSCHECK-BLOCK keyword prevents all users from running FSCHECK, and the FSCHECK-ALLOW-SM keyword will allow all users with SM capability to bypass this restriction.

If you need to restrict access to FSCHECK while allowing limited access, you can use the following $FILE-ACCESS section File protection rules instead of the $CONFIG section rules (which are shown here commented-out):

```
$CONFIG

  #     FSCHECK-BLOCK
  #     FSCHECK-ALLOW-SM
  #     FSCHECK-LOG

$FILE-ACCESS

    FILE=@.@.@
    TERM=20
         192.168.111.21
    USER=MANAGER.SYS
    ACTION=LOG

    FILE=FSCHK###.MPEXL.TELESUP
    TELL="FSCHECK unavailable"
    USER=@.@
    ACTION=NOOPEN,LOG
```

This rule works by blocking access to the message catalog file used internally by FSCHECK. It would also be possible to block the ability of users to run the FSCHECK program file, but this is not recommended as it would be trivial for an SM-capability user to bypass such a rule by copying FSCHECK to a different name or location (because FSCHECK still works when renamed).

The example also contains a rule that provides a controlled 'back-door': MANAGER.SYS can bypass all protection rules when logged on to LDev 20 or to a specific IP Address. Without this rule there would be no way to access FSCHECK.

Restricting Access to ORBIT'S BACKUP+

Orbit Software's Backup+ product uses low-level file-access mechanisms which bypass the documented file system calls intercepted by IDent. Under normal circumstances this is not a problem, but the Backup+ STORE command ;PURGE keyword bypasses file protections specified using the IDent $FILE-ACCESS section 'ACTION=PROTECT' or 'ACTION=NOPURGE' keywords, and allow the removal of critical files.

The BACKUPPL program file may be renamed, so the best way to restrict access to Backup+ is to restrict access to the IJGCONF.PUB.SYS license file that Backup+ verifies on startup.

The following example shows how access to Backup+ may be limited to specified users on a limited set of LDevs or IP Addresses:

```
$FILE-ACCESS

    FILE=IJGCONF.PUB.SYS
    TERM=20
         192.123.456.45
    USER=JOHN,MANAGER.SYS
         OPERATOR.SYS
    ACTION=LOG

    FILE=IJGCONF.PUB.SYS
    TELL="BACKUPPL unavailable"
    USER=@.@
    ACTION=NOOPEN,LOG
```

With these rules, JOHN,MANAGER.SYS and OPERATOR.SYS will be allowed access to BACKUP+ from LDev 20, or from the specified IP Address. All other users (both Jobs and Sessions) will be blocked.

Restricting Access to STORE's ";PURGE" Keyword

The :STORE command's ;PURGE keyword may be used to purge files immediately after storing them. IDent will prevent files from being purged this way when $FILE-ACCESS section protection rules include the PROTECT keyword.

The :RESTORE command may be used to restore files on top of existing files. IDent will prevent files from being overwritten this way when $FILE-ACCESS section protection rules include the PROTECT keyword.

Restricting Access to MPEX Commands

IDent protects files from unauthorized modification or removal by MPEX without any special configuration settings required. It is therefore not necessary to implement special restriction rules that target the MPEX utility.

NEWPASS= Keyword Syntax

The $DB-PASS section NEWPASS= keyword is used to specify the new Password or Password Class to be supplied if the rest of the Rule evaluates True. The NEWPASS= keyword may be used to specify a literal password, or (preferably) to specify a User Class number (by preceding the number with a '#').

When specifying a literal password, the password should be quoted if it contains any non-alphanumeric characters, or any lower-case letters (unquoted strings are up-shifted).

TurboIMAGE database User Class numbers lie in the range 1:63. IDent also allows the two special User Class numbers 0 and 64 to be specified, as described below.

NOTE: When using the NEWPASS= keyword to specify a User Class number, the '#' prefix character must immediately follow the preceding '=' character. If the '#' is preceded by a space it will be treated as a comment prefix, and the rest of the line will be ignored.

Special Interpretation of "NEWPASS=#64"

TurboIMAGE allows Users logged on as the creator of a database to supply the special ";" password, which guarantees full read/write access to all elements in a database. DBOPEN returns User Class 64 when the ";" password is used by the creator.

IDent allows $DB-PASS section rules to specify "NEWPASS=#64". When a rule containing this keyword evaluates True, IDent will automatically switch the user's logon to match that of the Database creator, and will then call DBOPEN with the ";" password supplied. Following the DBOPEN call, the user's logon will automatically switch back to their original logon.

There are two requirements for IDent's substitution of the ";" Password to work: the Database Creator must actually be defined on the system, and the Database must be located in the same Account as the Database Creator.

Special Interpretation of "NEWPASS=#0"

TurboIMAGE will allow a database to be opened using a null password if the database contains any unprotected datasets. DBOPEN returns User Class 0 when a null password is used, and will grant read access to all unprotected datasets.

IDent allows $DB-PASS section rules to specify "NEWPASS=#0", and will supply a null password when such a rule evaluates True.

Displaying TurboIMAGE User Class Numbers and Passwords

To display all User Class Numbers and Passwords defined for any TurboIMAGE database, use the DBUTIL utility "SHOW base PASSWORDS" command:

```
:DBUTIL

>> SHOW db PASSWORDS

  1 READER    2 WRITER    3           4           5
  6           7           8           9          10
 11          12          13          14          15
 ...
```

DBUTIL either requires you to be logged on as the database creator, or to have SM or AM capability to display currently defined passwords.

Debugging OPENFILE with Vesoft's MPEX

Effective use of the OPENFILE= keyword required knowledge of the files open by a given process. This can be hard to confirm; one way to display this information uses Vesoft's MPEX utility, and its enhanced %SHOWPROC command.

For example:

```
%SHOWPROC pin; FILES=@.@.@
```

In cases where a program is run with its STDIN redirected, the following syntax must be used to show all files (including the STDIN file):

```
%SHOWPROC pin; FILES=@.@.@; PRIV
```

Logging Visual Redo Edits

Vesoft's MPEX and Robelle's Qedit both implement visual-redo style REDO commands, where commands are edited using control-codes. In visual-redo mode, IDent logs every keystroke (including control-codes and spaces), allowing the edited command to be reconstructed.

This example shows SHOWOUT being edited to SHOWME inside MPEX:

```
22:06/0088/ %showout
22:06/0088/ %redo
22:06/0088/ <space>
22:06/0088/ <space>
22:06/0088/ <space>
22:06/0088/ <space>
22:06/0088/ ^D
22:06/0088/ ^D
22:06/0088/ ^D
22:06/0088/ m
22:06/0088/ e
```

Logging Input in Block-Mode Programs

By default, when a user is configured for input logging, any data typed inside block-mode programs will also be logged to the IDent Log File. If the CONFIG file NOLOG-BLOCK-MODE keyword is specified, data input in Block Mode will not be logged.

When block-mode data is logged, all logged data will be delimited by two lines that contain '===' characters. Block-mode logged data occupies the full width of the Log file, and is not prefixed by the timestamp that normally precedes every line in the Log file. The initial '===' line is time-stamped to identify when the screen was captured.

The format of logged block-mode screens depends on the program being run. Any data input into VPLUS-based block-mode programs will be logged as a series of '[]' delimited fields. Screen input fields are only logged if the user either types in them, or tabs over them. If a user 'tabs' over a field without typing in it (and even if the field displays data), the field will be logged as an empty '[]' field.

This example shows how data input to a VPLUS program is logged:

```
07:34/0090/ :nmmgr
07:34/0074/ ================================================================
[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa] [bbbbbbbbbbbbbbbbbbbb]
[ccccccccccccccccccccc] [] [eeeeeeeeeeeeeee]
================================================================================
07:35/0074/ <F8>
07:35/0090/ :bye
```

Fields are logged in the same order as on the displayed screen, but re-formatted to occupy the minimum number of lines in the Log file.

Data input using Robelle's Qedit editor in full-screen mode will be logged as the entire screen, as this example shows:

```
07:34/0090/ :qedit
07:34/0074/ /t config.pub.ptc
07:34/0074/ /vi
07:34/0074/ ================================================================

    Start of file  1 Text CONFIG.PUB.PTC "$"(1/72) Hint: Type ?f7 for Help
 *  ########################################################
+1  # IDent/3000 Configuration File
+2  #
+3  # hpsusan   :  12345678
+4  # hpcpuname :  SERIES 918LX
+5  # system2   :  1111-11-1111111111
+6  ########################################################
+7
+8  $CONFIG
+9
10  #   peutil-block
//  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7
================================================================================
07:35/0074/ <F8>
```

The Qedit logging code will accommodate screens of any length, as well as displays up to 132 characters wide.

$CONFIG Section TRACE-RULES Keyword

The TRACE-RULES keyword can be useful in determining why particular CONFIG file rules are or are not taking effect. When this keyword is specified, IDent creates a trace file in the TRACE.PTC group for each Job and Session, and writes diagnostic information in the file when it evaluates any rule in the CONFIG file $LOG-USER, $DB-PASS, $DB-ACCESS, or $FILE-ACCESS sections.

The diagnostic information shows how every keyword in every rule evaluates, and is intended to assist you in determining which rules are ultimately applied to user access attempts.

Note that IDent scans a compacted copy of CONFIG.PUB.PTC, where rules using USER= or TERM= keywords that evaluate False may have been eliminated. These rules will not be visible in the IDent trace file. For more details see **When CONFIG File Changes Take Effect** in Section 3.

Trace file names are derived using the same method that IDent uses for its log file: file names are 'S*nnnn*.TRACE.PTC' for Sessions, and 'J*nnnn*.TRACE.PTC' for Jobs, where *nnnn* is the Job or Session number.

For example, if the CONFIG file contains these rules:

```
$CONFIG

   TRACE-RULES

$LOG-USER

   USER= @.@:S
        LOG-DISCARD

$DB-PASS

   USER=@.@
   PASS="reader"
   NEWPASS=#10

   USER=@.@
   PASS="CREATOR"
   NEWPASS=#64
```

and the user attempts two DBOPENs using Query:

```
:query
HP32216N.03.15  QUERY/NM  FRI, MAR  7, 2008,  9:58 AM
COPYRIGHT HEWLETT-PACKARD CO. 1976

>b=testdb
PASSWORD = >>TEST
MODE = >>1

>b=testdb
PASSWORD = >>CREATOR
MODE = >>2
```

then these messages will be written to the trace file:

```
09:58/0077/ Scanning $LOG-USER section...
09:58/0077/
09:58/0077/  =  $LOG-USER    USER=     @.@:S
09:58/0077/  =  $LOG-USER    USER=     LOGDISCARD
09:58/0077/  => $LOG-USER    Rule #1    => Matches
09:58/0077/
```

```
09:59/0077/ Scanning $DB-PASS section...
09:59/0077/
09:59/0077/ =  $DB-PASS      USER=     @.@
09:59/0077/ -  $DB-PASS      PASS=     reader;
09:59/0077/ =  $DB-PASS      NEWPASS=  #10
09:59/0077/ -> $DB-PASS      Rule #1    -> no match
09:59/0077/
09:59/0077/ =  $DB-PASS      USER=     @.@
09:59/0077/ -  $DB-PASS      PASS=     CREATOR;
09:59/0077/ =  $DB-PASS      NEWPASS=  #64
09:59/0077/ -> $DB-PASS      Rule #2    -> no match
09:59/0077/
09:59/0077/ Scanning $DB-PASS section...
09:59/0077/
09:59/0077/ =  $DB-PASS      USER=     @.@
09:59/0077/ -  $DB-PASS      PASS=     reader;
09:59/0077/ =  $DB-PASS      NEWPASS=  #10
09:59/0077/ -> $DB-PASS      Rule #1    -> no match
09:59/0077/
09:59/0077/ =  $DB-PASS      USER=     @.@
09:59/0077/ =  $DB-PASS      PASS=     CREATOR;
09:59/0077/ =  $DB-PASS      NEWPASS=  #64
09:59/0077/ => $DB-PASS      Rule #2    => Matches
09:59/0077/
```

Annotations at the start of each line indicate if each keyword or rule evaluates True or False:

=	indicates that a keyword evaluates True
-	indicates that a keyword evaluates False
=>	indicates that a rule evaluates True
->	indicates that a rule evaluates False

The first DBOPEN attempt matches no $DB-PASS rules, the second DBOPEN attempt matches the second rule. $DB-PASS Section rules are scanned until the first match is found.

The IDent trace files in TRACE.PTC remain after a Job or Session logs off; they must be manually removed. IDent trace files are not archived or FTP'd automatically by IDent when a Job or Session logs off.

Once a Job or Session has written to its IDent trace file, the file will remain open until that Job or Session logs off. It it's necessary to remove an in-use trace file, the :PURGELINK command can be used.

The format of trace messages is not guaranteed to remain constant between IDent releases.

$CONFIG Section TRACE-DBOPEN Keyword

The $CONFIG section TRACE-DBOPEN keyword caused IDent to write additional debugging information to the IDent Job/Session trace file whenever DBOPEN is called. The debugging information assists in determining which password was provided to DBOPEN, whether IDent substituted a different password, and whether the DBOPEN call succeeded or not.

The first example shows that the user supplied password "TEST" from inside Query, and that DBOPEN succeeded, although it only granted User class 0 access to the database. This indicates that the supplied password does not match any password in the database, but that the database contains unsecured Fields that allow access without supplying a valid password:

```
12:49/0022/    DBOPEN call summary ----------------
12:49/0022/    Base          TESTDB.DB.DEV
12:49/0022/    Owner         MGR.DEV,DB
12:49/0022/    User          PAUL,MGR.DEV
12:49/0022/    Prog          QUERY.PUB.SYS
12:49/0022/    Prog:1        CI.PUB.SYS
12:49/0022/    Mode          1
12:49/0022/    IP            192.168.111.5
12:49/0022/    UDC           LOGON
12:49/0022/    Pass          "TEST;"
12:49/0022/
12:49/0022/    DBOpen returned status: 0; class: 0
```

The next example show that an IDent rule has triggered, and that the rule contains keyword "NEWPASS=#64". This causes IDent to temporarily switch the User's logon to that of the database creator and substitute the ";" password. The User's logon is then returned to its original location. The trace shows that the DBOPEN call succeeded with User class 64 access granted.

```
12:49/0022/    Switch logon to MGR.DEV,DB
12:49/0022/
12:49/0022/    DBOPEN call summary ----------------
12:49/0022/    Base          TESTDB.DB.DEV
12:49/0022/    Owner         MGR.DEV,DB
12:49/0022/    User          PAUL,MGR.DEV
12:49/0022/    Prog          QUERY.PUB.SYS
12:49/0022/    Prog:1        CI.PUB.SYS
12:49/0022/    Mode          2
12:49/0022/    IP            192.168.111.5
12:49/0022/    UDC           LOGON
12:49/0022/    Pass          "CREATOR;"
12:49/0022/    Newpass       ";" class:64
12:49/0022/
12:49/0022/    Switch logon to PAUL,MGR.DEV,PUB
12:49/0022/
12:49/0022/    DBOpen returned status: 0; class: 64
```

TRACE-DBOPEN can be used in conjunction with the TRACE-RULES keyword (described above) to determine which $DB-ACCESS rule triggered, and the effect it had on the DBOPEN call.

When TRACE-DBOPEN is specified, IDent trace files contains clear-text TurboIMAGE passwords; they should therefore be purged as soon as the CONFIG rules have been debugged, and the TRACE-DBOPEN keyword should only be used when necessary.

Notes on $FILE-ACCESS Protection and File System Calls

IDent implements file access riles defined in the CONFIG file $FILE-ACCESS section using interception logic that can modify file open and close calls made by application programs as well as by MPE/iX itself, to prevent files being accessing in ways that have been blocked. This is necessary because many combinations of [HP]FOPEN and FCLOSE parameters can be used to cause file contents to be deleted or overwritten, and IDent must prevent this occurring.

Depending on the exact $FILE-ACCESS protection rules defined, and the parameters that a program uses to attempt to open or close a file, any of the following messages may be generated by IDent:

- Blocked destructive Read access to
- Blocked Write-Copy access to
- Blocked Open of
- Blocked Read access to
- Blocked Append access to
- Blocked Execute access to
- Blocked Overwrite access to
- Blocked Write access to
- Blocked Update access to
- Blocked Purge of
- Blocked MakeTemp of
- Blocked Rename to
- Blocked Rename of
- Blocked Save of
- Blocked access to

- Allowed Read access to
- Allowed Write access to
- Allowed Write(save) access to
- Allowed Append access to
- Allowed Read/Write access to
- Allowed Update access to
- Allowed Execute access to
- Allowed All access to
- Allowed Read access to
- Allowed Save of

These messages will be logged by IDent, according to the relevant ALLOW-TO-XXX and BLOCK-TO-XXX settings in the CONFIG file $CONFIG section.

Modification of File Equation Keywords

Some File Equation keywords could be used to subvert the ability of IDent to implement all of the file access protection rules defined in the CONFIG file $FILE-ACCESS section. Therefore IDent blocks attempts to use the :FILE command ;OUT=to establish File Equations for protected files that contain any of the following keywords:

```
;ACC=OUT
;ACC=OUTKEEP
;COPY
;NOCOPY
```

IDent logs attempts to use these keywords on protected files, and modifies relevant :FILE commands so that just these keywords are removed; all other legal keywords will be applied.

As File Equations may specify partially-qualified file names, it's possible for a legal File Equation to become illegal after the user's logon is changed using the MPEX %CHLOGON command (or any other logon-switching program). For this reason, all File Equations are re-scanned when %CHLOGON is used, and any illegal keywords will be automatically removed at that time.

Limitations on use of $FILE-ACCESS Section ACTION=NORUN Keyword

Although the $FILE-ACCESS section "ACTION=NORUN" keyword may appear to be a good way to create protection rules that limit access to program files, it is actually only a good solution when used to protect program files that contain checks to prevent themselves being renamed or copied to different names or locations. An SM capability user can always copy a program to a new name, and hence a protection rule that checks the name of the program is not reliable. This keyword may also prove useful in limiting the ability of low-capability users to run programs that they would normally only have execute access to.

This keyword can be used to protect the IDent program OFF.PUB.PTC, because that program will not run if moved to a different location.

Protecting IDent Files

It is important for the integrity of the audit trail that all IDent Log Files are protected against removal or tampering. It’s also crucial to protect the various files that allow IDent to run from malicious tampering.

Here is a suggested method to define suitable protection, allowing access to one user when logged on in a secure location:

```
$FILE-ACCESS

    FILE=@.@.@
    TERM=123
         122.320.122.5
    USER=JOE,MANAGER.SYS
    ACTION=LOG

    FILE=@.@.PTC
    ACTION=PROTECT,LOG
```

Similar rules should be used for any other log files that must be protected against deletion.

CI Error Reporting with Protected Files

When attempting to use MPE or MPEX commands to Purge files which are protected by $FILE-ACCESS section rules, you may see situations when it appears that the files have been purged. This is due to technical limitations which make it impossible to pass a suitable status back to the active program (in this case CI.PUB.SYS or MAIN.PUB.VESOFT) at FCLOSE time. IDent prevents the files from being purged, even though the active program does not see an error status when it attempts to purge the files.

If the files are protected by a $FILE-ACCESS section rule that specifies ACTION=LOG, and if the CONFIG LOG-TO-TERM keyword is specified, then the user will see a message that indicates that the Purge operation was blocked.

$CONFIG Section LOG-CHLOGON Keyword

Some programs (FTPSRVR being a good example) temporarily change their logon ID to a different location than the Job/Session in which they run. In the case of FTPSRVR, this allows the FTP server process inside the JINETD Job to respond to incoming connections to different logon IDs. Vesoft's MPEX implements a %CHLOGON command which allows a form of temporary nested logon. In both cases logon switching is accomplished by calling the system AIFCHANGELOGON intrinsic.

A single process may change its logon ID multiple times during its lifetime, switching between many different logon IDs. A change in logon ID may be 'global' (in which case SHOWJOB will reflect the new logon ID), or 'process-local', in which case the Job or Session logon ID does not

change. It may not be apparent when a process performs a process-local change to its logon ID, and operates under a different logon ID to the Job or Session in which it runs.

The $CONFIG Section LOG-CHLOGON keyword will cause IDent to log all calls that change a program's logon ID, both global and process-local. The following log file fragment from the JINETD Job's IDent log file shows the result of two different incoming FTP connections:

```
12:31/0182/ :RUN inetd.net.sys
13:43/0178/ Process-Local CHLOGON to XAVIER,MGR.DEV ---------------------------
13:43/0178/  via FTPSRVR.ARPA.SYS via INETD.NET.SYS ---------------------------
13:43/0178/
13:45/0187/ Process-Local CHLOGON to OPERATOR.SYS -----------------------------
13:45/0187/  via FTPSRVR.ARPA.SYS via INETD.NET.SYS ---------------------------
13:45/0187/ Switch to Privileged logon with Caps=SM,OP,PM ---------------------
```

One incoming connection connects as XAVIER,MGR.DEV using pin 178, a second incoming connection connects as OPERATOR.SYS using pin 187. Both calls are process-local, meaning that :SHOWJOB continues to show that the Job is logged on as JINETD,MANAGER.SYS. The $CONFIG Section LOG-VIA keyword is also active in this example, showing the path by which CHLOGON is called.

When IDent's $FILE-ACCESS and $DB-ACCESS rules are used to limit file access as a function of a user's logon ID, IDent uses the global Job or Session logon ID to evaluate USER= keywords in rules when determining if access is allowed or blocked; it does not use whatever process-local logon ID a process temporarily runs as.

Logging Incoming FTP File Accesses

If it is desired to log all file accesses performed by incoming FTP connections, a rule similar to the following could be used, preferably placed after all other $FILE-ACCESS Section rules:

```
$FILE-ACCESS

   file=@.@.@                                  # trace FTP activity
   user=jinetd,manager.sys:J                   #
   prog=ftpsrvr.arpa.sys                       #
   action=log                                  #
```

The system FTP Server uses the program file FTPSRVR.ARPA.SYS, run from inside the Job JINETD.ARPA.SYS. This rule will trigger when the FTP Server is run, and results in all allowed file accesses being logged to the IDent log file corresponding to the JINETD Job.

The following log file fragment from a JINETD Job's IDent log file shows how various FTP GET and PUT commands will be logged:

```
12:31/0182/ :RUN inetd.net.sys
13:43/0178/ Allowed Append access to FTPLOG.ARPA.SYS -------------------------
13:43/0178/ Allowed Read access to FTPC000.ARPA.SYS --------------------------
13:43/0178/ Process-Local CHLOGON to XAVIER,MGR.DEV --------------------------
13:45/0187/ Allowed Write(save) access to FILE01.PUB.DEV ---------------------
13:45/0187/ Allowed Read access to FILE02.PUB.DEV ----------------------------
13:47/0187/ Allowed Save access to FILE03.PUB.DEV ----------------------------
```

Note that as the $CONFIG section rule specifies "File=@.@.@", all file accesses made by FTPSRVR are logged, including calls to FTPLOG and FTPC000 (FTP's log file and message catalog).

Depending on the actual sequence of file system intrinsic call made by FTPSRVR, files being sent using the PUT command may be identified by "Allowed Write(save) access" or "Allowed Save access" messages; files retrieved using the GET command will be identified by "Allowed Read access" messages.

Identifying Logons where :HELLO ;PARM=1 or -2 is specified

MPE/iX allows SM capability users to logon with :HELLO; PARM=-1 or -2 specified. When logged on this way all UDCs are bypassed, including logon UDCs that may have been used to implement logon security checks.

IDent will, by default, log all logons that use :HELLO ;PARM=-1 or -2 to bypass UDCs. Such logons create the following message (shown in bold) in the IDent log file:

```
17:47/0091/ ====================================================================
17:47/0091/ IDent/3000  (c) Paul Taffel Consulting (PTC), 2001-2010
17:47/0091/
17:47/0091/ #S5457 logged on as MGR.DEV
17:47/0091/ LDev 8; IP Address 192.168.111.6
17:47/0091/ SUN, MAR 28, 2010  5:47 PM
17:47/0091/
17:47/0091/ Keywords: LOG-DISCARD, LOG-PRIV-BOOST, LOG-PRIV-LOGON.
17:47/0091/ ====================================================================
17:47/0091/
17:47/0091/ Session UDCs nullified using :HELLO ;PARM=-1 ----------------------
17:47/0091/
17:47/0091/ Privileged logon with Caps=SM,PM ---------------------------------
```

When the CONFIG file's LOG-DISCARD keyword is active, this log message will cause the log file to be retained.

It's possible to close the -ve PARM loophole system-wide, by using SYSGEN's enforcelogonudcs setting (in the MISC section).

Vesoft's Security/3000 logon security can also be configured (using %BACKG LOGON) so that its logon security checks do not rely on UDCs being set.

Security/3000's %BACKG HELLO command can be configured so that attempts to specify -ve PARM values are stripped out from the :HELLO command. See Vesoft's documentation on the $NOPARM keyword for more information. If $NOPARM is used, -ve PARM values will be removed before IDent processes the HELLO command, and hence this IDent log message will not be generated.

Section 16: IDent Solutions to PCI DSS 1.2 Requirements

This section summarizes the specific Payment Card Industry Data Security Standard (PCI DSS) version 1.2 requirements that can be addressed using IDent, and the IDent rules that contribute to each solution.

PCI DSS Requirement		IDent Solution
2.1	Don't use vendor-supplied defaults for system passwords and other security parameters.	$DB-PASS Section rules allow database User access to be defined independently of database passwords. Once rules are defined, database passwords can be changed (using DBUTIL or 3rd party tools) whenever needed, without requiring any user program changes.
2.4	Shared hosting providers must protect each entity's hosted cardholder data.	$USERSET Section rules can be used to define lists of users in each entity. $DB-ACCESS and $FILE-ACCESS rules with USER= keywords can reference these user lists, and can be used to limit cardholder data access to authorized users.
3.3	Mask PAN when displayed.	$CONFIG Section CC-MASK and CC-MASK-ALL keywords cause any Credit Card numbers written to IDent log files to be partially or fully masked.
10.2	Implement automated audit trails for all system components to reconstruct...	$LOG-USER Section rules allow collection of complete Job/Session audit trails. These audit log files allow reconstruction of user actions before and after events that are flagged in IDent log files.
10.2.1	...individual access to cardholder data.	$DB-ACCESS Section rules using the SETS= and ACTION= keywords allows rules to be constructed that specify the access allowed to datasets that contain cardholder data, as well as whether access attempts should be logged. The $FILE-ACCESS Section provides similar protection for any flat files that contain card-holder data. The $CONFIG Section ALLOW-TO-LOG and BLOCK-TO-LOG keywords will cause access attempts to be flagged in the IDent log file. $LOG-USER Section rules combine these access flags with all user input in a single log file.
10.2.2	...all actions taken by any individual with root or administrator privileges.	The $LOG-USER Section keywords LOG-PRIV-LOGON and LOG-PRIV-BOOST can be used to flags users who logon with administrator privileges, or who acquire them after logon. SM or PM capabilities are treated as privileged. The $CONFIG Section OP-IS-PRIV keyword can be used to include OP capability users as privileged. $LOG-USER Section rules combine the privilege flags with all user input in a single log file.

PCI DSS Requirement		IDent Solution
10.2.3	...access to all audit trails.	$FILE-ACCESS Section rules using the ACTION= keyword allows rules to be constructed that specify the access allowed to IDent log files (or other audit trail files), as well as whether access attempts should be logged. The $CONFIG Section ALLOW-TO-LOG and BLOCK-TO-LOG keywords will cause access attempts to be flagged in the IDent log file. $LOG-USER Section rules combine these access flags with all user input in a single log file.
10.2.4	...invalid logical access attempts.	The $CONFIG Section BLOCK-TO-LOG keyword causes access-blocked messages to be flagged in the IDent log file when any $DB-ACCESS or $FILE-ACCESS Section rules block access to a restricted dataset or flat file.
10.2.6	...initialization of the audit logs.	IDent audit log files are created automatically for all Jobs/Sessions when matching $LOG-USER rules are in place. Console messages are generated when IDent is enabled or disabled, and the $CONFIG Section keyword STATUS-TO-MONITOR can be used to copy these status messaged to a reserved MONITOR Session.
10.2.7	...creation and deletion of system-level objects.	IDent log files contain all user input, regardless of what program is used to prompt for input. Every IDent log file line is timestamped, and contains a copy copy of the active prompt when input is entered. This information allows reconstruction of all user actions in a given Job or Session.
10.3	Record at least the following audit trail entries for all system components for each event:	IDent log files contain a copy of everything typed by the user being monitored. User input is logged together with the displayed prompt, to aid in interpreting the audit trail. In addition...
10.3.1	User identification.	...IDent includes user information at the start of each log file, identifying logon user name, LDev number or IP Address, and streaming user (for Jobs).
10.3.2	Type of event.	...IDent log file messages identify the event type.
10.3.3	Date and time.	...IDent log file messages identify the date and time the event was logged.
10.3.4	Success or failure indication.	...IDent log file messages indicate if the logged event was allowed or blocked. The $CONFIG Section keyword LOG-CIERROR can be used to copy the text of CI error messages to the log file.

PCI DSS Requirement		IDent Solution
10.3.5	Origination of event.	...The context of IDent log messages identifies the program or command used. The $CONFIG Section keyword LOG-VIA will append to all event messages the chain of commands, programs, command files, or UDCs used prior to event detection.
10.3.6	Identity or name of affected data, system component, or resource.	...IDent log messages identify the name of the resource being allowed or blocked.
10.5	Secure audit trails so they cannot be altered.	$FILE-ACCESS Section rules that cover the location of audit log files with the ACTION=PROTECT keyword specified will prevent tampering with the audit log files.
10.5.1	Limit viewing of audit trails to those with a job-related need.	$FILE-ACCESS Section rules with the ACTION=NOREAD keyword can be used to limit read access to IDent log files to authorized users.
10.5.2	Protect audit trails from unauthorized modifications.	$FILE-ACCESS Section rules with the ACTION=PROTECT keyword can be used to prevent tampering with IDent's log files.
10.5.3	Promptly back up audit trail files to a centralized log server or media that is difficult to alter.	The $CONFIG Section FTP-START keyword (and its associated configuration keywords) create IDent's background FTPMGR Job. This job automatically sends IDent log files to a protected FTP server as soon as each monitored Job/Session logs off. IDent also moves individual Job/Session log files to the ARCHIVE.PTC group, which can be protected against tampering using $FILE-ACCESS Section rules.
10.5.5	Use file-integrity or change-detection software on logs to ensure that existing log data cannot be changed without generating alerts.	The $FILE-ACCESS Section rules guarantee that logs cannot be tampered with – if it is absolutely necessary to verify that archived copies of the log files on the HP3000 are also unmodified, the $CHECKSUM Section can be used to periodically verify that the files are unmodified.
11.5	Deploy file-integrity monitoring software to alert personnel to unauthorized modification of critical system files, configuration files, or content files, and configure the software to perform critical file comparisons at least weekly.	$CHECKSUM Section rules allow definition of filesets that contain 'critical' files. IDent's CHECKSUM utility scans the contents of these files and derives a cryptographic checksum for each file. CHECKSUM uses the knowledge of whether a file's checksum has changed to identify new, changed, or deleted files. The CHECKSUM utility can be run as often as needed to meet PCI-DSS requirements.

PCI DSS Requirement		IDent Solution
A-1	Shared hosting providers must protect the cardholder data environment.	See solution for requirement 2.4.
A.1.1	Ensure that each entity only run processes that have access to that entity's cardholder data environment.	See solution for requirement 2.4.
A.1.2	Restrict each entity's access and privileges to own cardholder data environment only.	See solution for requirement 2.4.