SECURITRE

Administrator Guide

This document is applicable to the SECURITRE Version 3.3.1 software package.

Comments pertaining to this document and the SECURITRE package are encouraged. Please direct all comments to:

Treehouse Software, Inc.
409 Broad St., Suite 140
Sewickley, PA 15143

phone: (412) 741-1677
fax: (412) 741-7245
e-mail: tsi@treehouse.com
http://www.treehouse.com

Worldwide marketing of SECURITRE and other products of Treehouse Software, Inc. (TSI) is handled through the Sewickley office.

Last Updated: 6/27/2008

* In this document, CA-ACF2 is referred to as ACF2, and CA-TOP SECRET is referred to as TOP SECRET or TSS.

TABLE OF CONTENTS

LIST OF FIGURES

This page intentionally left blank.

SECTION I

INTRODUCTION

I.1 SECURITRE Documentation

The structure of the SECURITRE documentation is intended to make information about the product more convenient to locate and use.

Treehouse Software, Inc. (TSI) provides two manuals for SECURITRE. In order to successfully install and use SECURITRE, both manuals are required.

Administrator Guide
The Administrator Guide provides detailed explanations for installing, setting up, and tailoring SECURITRE to site-specific needs.

The Administrator Guide explains how to install and prepare SECURITRE for use by using a simple, efficient process. It explains installation details for SECURITRE modules and adjustments necessary in RACF, ACF2, and TOP SECRET. It also explains the few primary parameters and modules necessary to get SECURITRE running in a TEST environment, giving several comprehensive examples. Once SECURITRE is operational, it is necessary to view the SECURITRE Reference Manual for information about placing SECURITRE into PRODUCTION, fine tuning, and defining less frequently used parameters.

Reference Manual
The Reference Manual provides detailed reference material about the various SECURITRE functions and features.

The Reference Manual is intended for reference use after the product has been installed and its successful operation has been verified. The Reference Manual lists and describes items that are typically referenced. Parameters are listed in alphabetical order, and Error Codes are listed in numeric order. There is little introductory discussion in this manual.

The nature of security precludes giving detailed security information or describing security techniques to all except those with an absolute need for the information. Therefore, there is no "User Manual" with SECURITRE. The “users” in this case are end-users and applications programmers. These personnel do not normally need to know if security is in effect or how it is employed. They only need to know that they should report to their management if they receive a security violation message.

Those people with a need for SECURITRE information include:

- The highest level of management to be assured that their data and applications are secure
- The Security Administrator, Auditors, and the Security Staff
- The DBA and/or DBA Staff
- The NATURAL Administrator(s)
- The System Programmers and Operations Staff for installation help
- The Applications Analysts and Project Leaders to understand SECURITRE's Application Security features (STRNAT and STRASM)

Additional documentary materials of value to SECURITRE sites are available from TSI free of charge. These include:

- Product Overview
- Fact Sheet

The following sections are presented in this Administrator Guide:

- Setting Up SECURITRE for ADABAS
- Setting Up SECURITRE for NATURAL
- SECURITRE for ADABAS Utilities
- Real-Time Monitor
- Internal Application Security Features
- User-Exits to SECURITRE
- User-Exit Co-existence
- Installation
- Operations Considerations

Note:	This Administrator Guide contains a Glossary of Terms that is located before the Index.

I.2 Version Enhancements

Note:	This release of SECURITRE supports NATURAL version 3.1.6 and above only. This is the last release of SECURITRE that will support NATURAL version 3.1.6. SECURITRE no longer supports CICS versions prior to 3.2. All versions of CICS Transaction Server (CICSTS) are support.

Note:	New modules, TSIRDC31, TSIRDC41, and TSIRDC42 are supplied with this release and must be used instead of TSIRDC when linking NATURAL. Refer to **Section IX.6.6 Link-edit a NATURAL Nucleus** of the SECURITRE Administrator manual for instructions on linking NATURAL.

Enhancements in this version include the following:

- NATURAL 4.2 support
- The zap status of SECURITRE Utility Security and SECURITRE's User-Exit-1 is now printed to the STRMSG DD if STRMSG is specified in the job.

Expiration Date /Authorization Zap

SECURITRE is distributed as an "Expired Trial". Sites installing SECURITRE must apply an Authorization Zap for "sold" status or an Expiration Zap for "trial" status. The same sold or trial zap used for Version 3.2 may be used for 3.3. If further instructions are necessary, please contact support@treehouse.com.

I.3 Introducing Central Security Management

Most IBM and IBM compatible mainframe installations rely on one of the three major operating System Security Facilities: RACF, CA-ACF2, or CA-TOP SECRET to control access to their non-ADABAS data and non-NATURAL applications. These System Security Facilities (SSFs) provide centralized security administration for all of these types of applications and datasets available on the computer system.

Centralized comprehensive control of the security function is essential to promoting the integrity and the safety of the computerized applications. Centralized control and administration is the most productive and cost-effective approach. The task of controlling a data center's resources is complicated when the Security Administrator must become involved with multiple disjointed security packages to control access to selected portions of a system, as is the case in typical ADABAS/NATURAL sites that do not use SECURITRE.

ADABAS and NATURAL do not interface directly with these centralized SSFs. Instead, ADABAS and NATURAL have their own security mechanisms.

ADABAS has password-based File Level Security, Field Level Security, and Security By Value (SBV) to control access to ADABAS files. However, the growth and diversity of information systems have made it nearly impossible to create and manage passwords effectively. In spite of an organization's policies and procedures, passwords often become common knowledge, and password protection is often circumvented so that the organization's mission can be accomplished unimpeded.

The NATURAL Security System (NSS) is an optional package that controls access to NATURAL, individual applications, and programs. Since NSS only controls NATURAL-based access of ADABAS data, ADABAS files (DDMs) have accessibility checks made by NSS, but only at the time that NATURAL programs are compiled. Therefore, NSS controls access to NATURAL and its applications and programs, but it does not secure data. Furthermore, Direct Calls, ADASQL, and other types of ADABAS data access are not secured by NSS.

Each of these ADABAS and NATURAL security systems or packages has its own administrative mechanisms, further complicating the task and decreasing the ability to adequately secure the data center. This increases the potential for security breaches or fraud. The auditors for many large organizations do not permit securing of the data and applications of data centers with tools or software provided by the database and application vendors because this adds to the potential risk. Furthermore, most auditors do not permit homegrown security-related software exits, hooks, etc.

Although these ADABAS and NATURAL security mechanisms can still be used, SECURITRE provides more precise and effective centralized control over ADABAS data and NATURAL applications. As a result, data and applications are more secure from unauthorized use or tampering.

SECURITRE acts as a software interface to RACF, ACF2, and TOP SECRET, allowing sites to merge the ADABAS/NATURAL security function into the standard SSF rule base. This means that security rules are maintained in one place for ADABAS and non-ADABAS data, along with NATURAL and non-NATURAL applications. The single rule base provides better security and makes it easier to manage changes in the security environment.

The process of controlling access to ADABAS and NATURAL is simplified and centralized. Control becomes more effective and training time is greatly reduced when a single, familiar mechanism is used to control both ADABAS and non-ADABAS environments and both NATURAL and non-NATURAL applications.

SECURITRE and the SSF control access to ADABAS data and NATURAL environments at all levels, diminishing or eliminating the need for separate ADABAS and NATURAL security control mechanisms, separate security files, and application-based security logic.
Under SECURITRE, data and applications security is based on User-IDs rather than passwords. Security is more precise and effective.

Changes in processing rules and requirements do not normally require changes to SECURITRE, only to the SSF database. This makes implementation of SECURITRE a relatively "painless" process.

Consider the environments on the following pages.

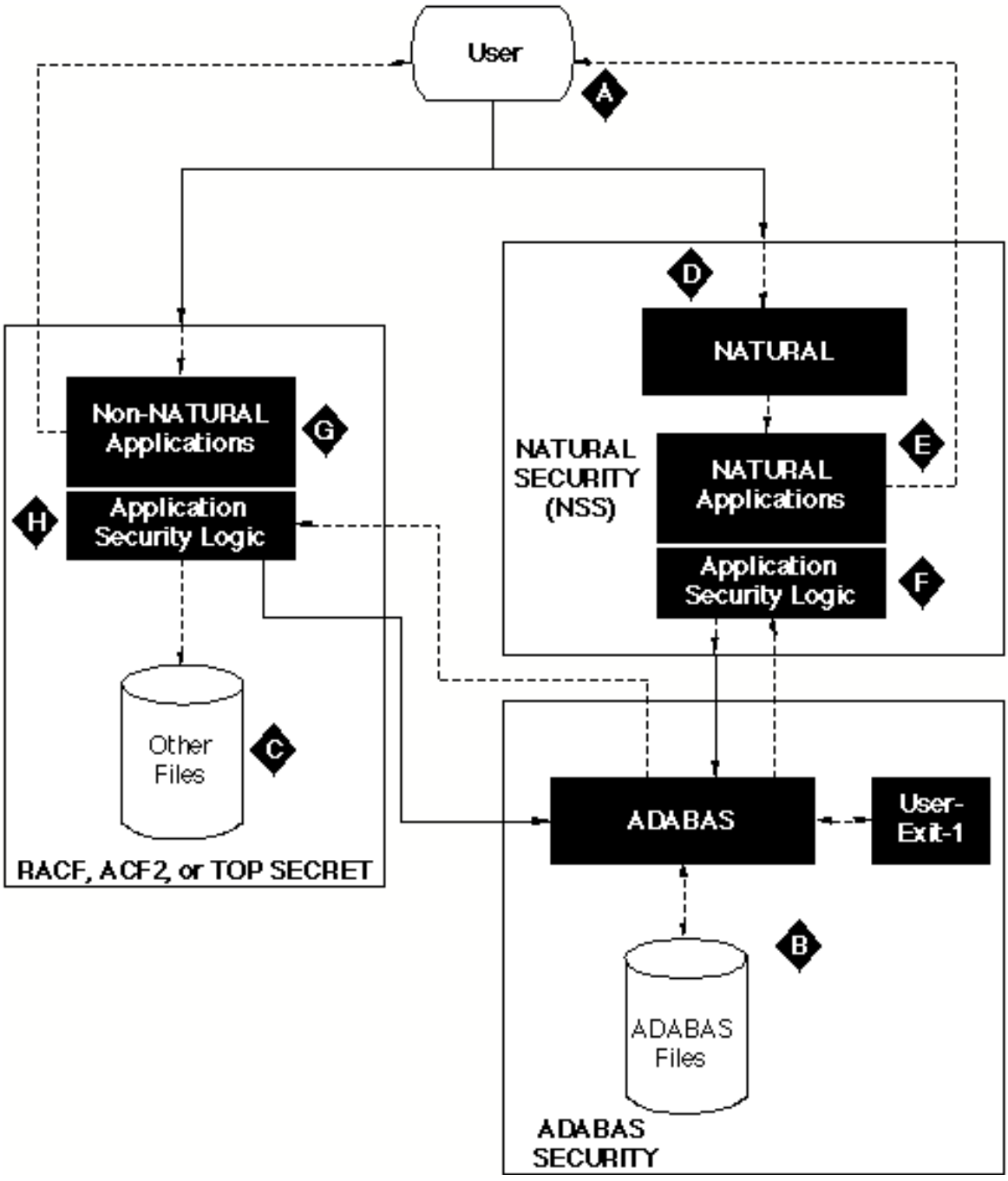

Figure 1 – ADABAS/NATURAL Control and Security without SECURITRE

In the diagram, the User **(A)** may access ADABAS files **(B)** and other (non-ADABAS) files **(C)**. Each of these comes with its own set of security implications in a typical system.

Before the User **(A)** uses NATURAL **(D)** to access ADABAS, the user may have to pass through the NATURAL SECURITY System (NSS). Access to NATURAL may be inhibited, or the NSS may disallow execution of the application **(E)** by this particular user. Access to ADABAS may be restricted by security logic **(F)** embedded in the NATURAL application software itself, by NSS invalidating certain programs, commands, or statements, or by NSS blocking access to certain databases or files (at compile time).

If access has not been stopped prior to this point, commands are issued to ADABAS **(B)** to retrieve the appropriate data. This retrieval may be inhibited by User-Exit-1 logic (TRIM, TSI's Performance Monitor, provides such logic as an option) or by ADABAS File/Field Level Security or Security By Value (SBV). If the retrieval is permitted, the requested data is returned by ADABAS to the NATURAL application program **(E)**. Embedded routines in the NATURAL program **(F)** may further limit the user's ability to process the information retrieved from ADABAS.

For non-NATURAL applications **(G)**, the SSF may prevent the user **(A)** from accessing the non-NATURAL applications. The user's access to certain data may be limited by security logic **(H)** embedded in the application software. Calls to retrieve non-ADABAS data are well controlled by the SSF. However, calls issued to ADABAS **(B)** may be halted by User-Exit-1 logic, ADABAS File/Field Level Security, or Security By Value (SBV). Additional routines in the application software **(H)** may limit the user's ability to process the information retrieved from ADABAS.

TSI developed SECURITRE to integrate ADABAS/NATURAL Security and Control with the overall system security provided by RACF, ACF2, or TOP SECRET.

With SECURITRE in place, the process of controlling access to ADABAS and NATURAL is simplified and centralized. The SSF controls access to ADABAS data at all levels, eliminating the need for separate ADABAS and NATURAL security control mechanisms and application security files. The following diagram depicts the new security arrangement.

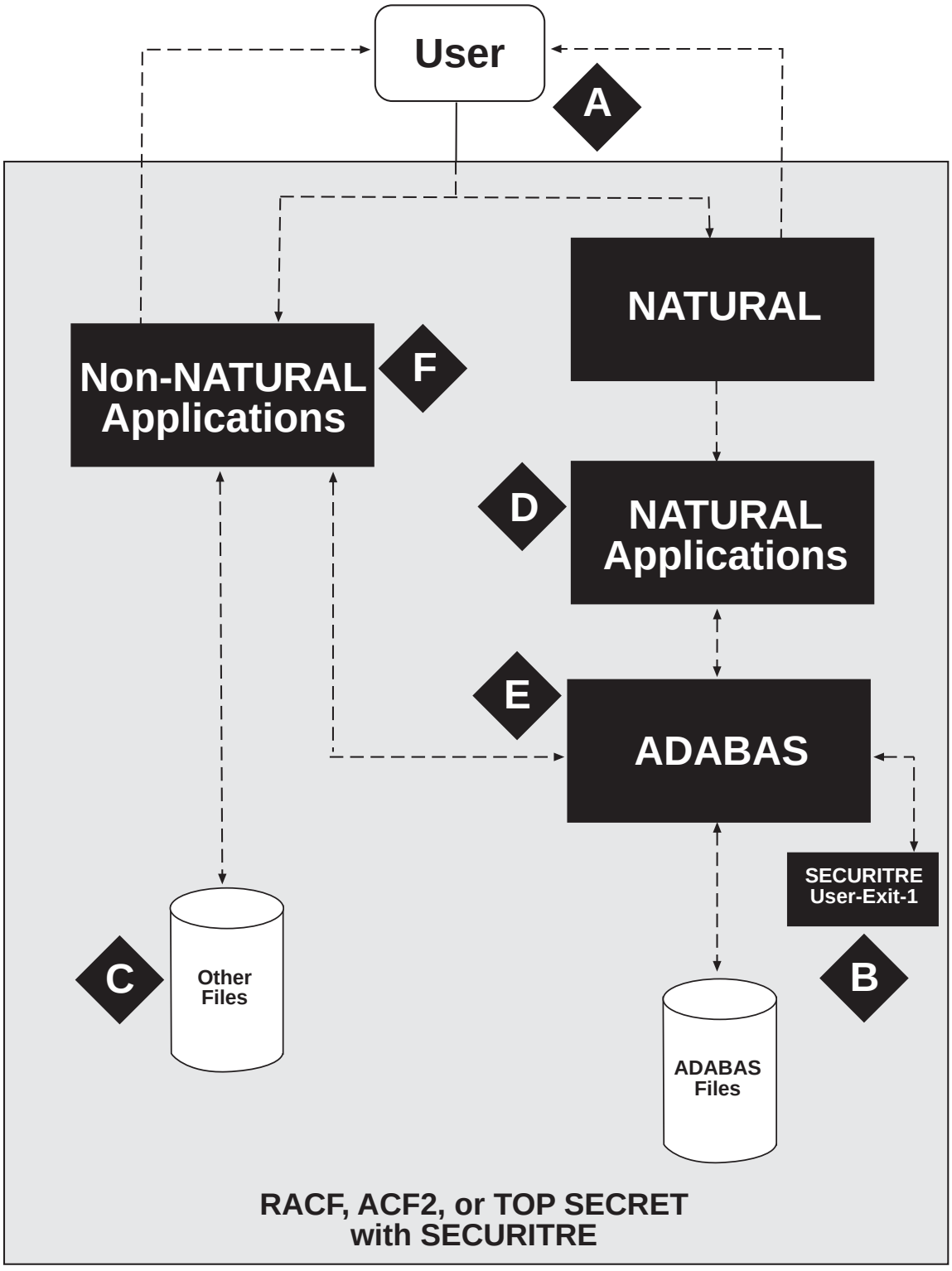

Figure 2 – ADABAS/NATURAL Control and Security with SECURITRE

Under SECURITRE, a user wishing to access ADABAS data may still do so through NATURAL or non-NATURAL application software. If the user chooses NATURAL application software, the SSF will determine if the user is allowed to access NATURAL **(D)** and its related system files (FNAT, FUSER, and FDIC) through SECURITRE. If allowed, the user will only be permitted access to approved applications (Libraries) **(E)** under SECURITRE control. A user's ability to Edit, Stow, or Execute may be restricted. This restriction can be applied at the program level. Furthermore, users may be inhibited from compiling programs that refer to restricted DDMs. The application may contain embedded security logic; further restricting access to the data, but this embedded security logic* can often be eliminated.

NATURAL issues the appropriate calls to ADABAS **(B)**. If SECURITRE allows the user to access the ADABAS file and fields in the desired manner, SECURITRE allows ADABAS to process the call and return the data to the NATURAL application **(E)** through the SSF. The NATURAL application's logic* may choose whether or not to pass the information on to the user **(A)**.

The process is similar if the user runs a non-NATURAL application **(F)**. If the SSF allows access to the non-NATURAL application, calls to ADABAS are controlled by SECURITRE through the SSF. Access to specific information may be further limited by security logic* embedded within the non-NATURAL application software.

ADABAS Utility execution is also controlled by SECURITRE. For more information, refer to ***Section IV.2 - ADABAS V5 and V6 Utility Control*** in the ***SECURITRE Reference Manua****l*.

* Without SECURITRE, embedded security logic may make decisions such as "can the particular user have access to a particular screen, have access at a particular time of day, access particular fields, or view data when fields have certain values?" STRNAT and STRASM can replace most embedded logic to help put these controls in the SSF.

I.4 The Functions of SECURITRE

SECURITRE consists of the following components:

- SECURITRE for ADABAS Nucleus modules
- SECURITRE for ADABAS Utilities modules
- SECURITRE for NATURAL modules
- Real-Time Monitor
- STRNAT and STRASM modules

The first three components mentioned above are the primary components of SECURITRE. SECURITRE for ADABAS Nucleus modules must be used, but the other two components are optional. Organizations typically implement them in the order shown above.

SECURITRE for ADABAS Nucleus

This function provides security for:

- ADABAS databases
- Files in each database
- Fields in each file

SECURITRE for ADABAS Utilities

This function provides control over use of:

- Utility programs
- Utility functions

For each:

- Database
- File

SECURITRE for NATURAL

This function provides control over:

- NATURAL session initialization for the DBID, FNAT, FUSER, FDIC combination
- Applications (i.e., private, public, or restricted libraries)
- All or selected programs for EXECUTE, EDIT, SAVE, STOW, and RUN
- DDM viewing, changing, and referencing in programs

Each of these three components communicate authorization requests to the SSF through use of a "pseudo-dataset name" to which a user may be granted access by the SSF. These requests are transparent to application programmers and end users. Note that the pseudo-dataset name does not refer to a physical dataset, but is merely a label used to identify a resource to be secured, such as a NATURAL DDM or an ADABAS file.

SECURITRE determines how to generate the pseudo-dataset name based on the parameter values specified by the site. For example, the Security Administrator may define file 131 on database 243 as "ADABAS.PROD.REVENUE". The ADABAS Utility ADADBS function for file 3 may be defined as "ADABAS.UTIL.DBS.RELEASE.F003".

The definition of pseudo dataset names and other definitions and variations are provided to SECURITRE through a set of flexible macro-generated parameters.
SECURITRE operates in any MVS environment and supports batch and commonly used teleprocessing systems, including TSO, CICS, and COM-PLETE.

SECURITRE has minimal impact on system response time. Through the use of efficient table mechanisms, SECURITRE can be placed into production environments without any question of performance.

SECURITRE is easy to install and use because it is reliable, efficient, and comprehensive.

The figure below illustrates how the SECURITRE modules fit into a site's environment:

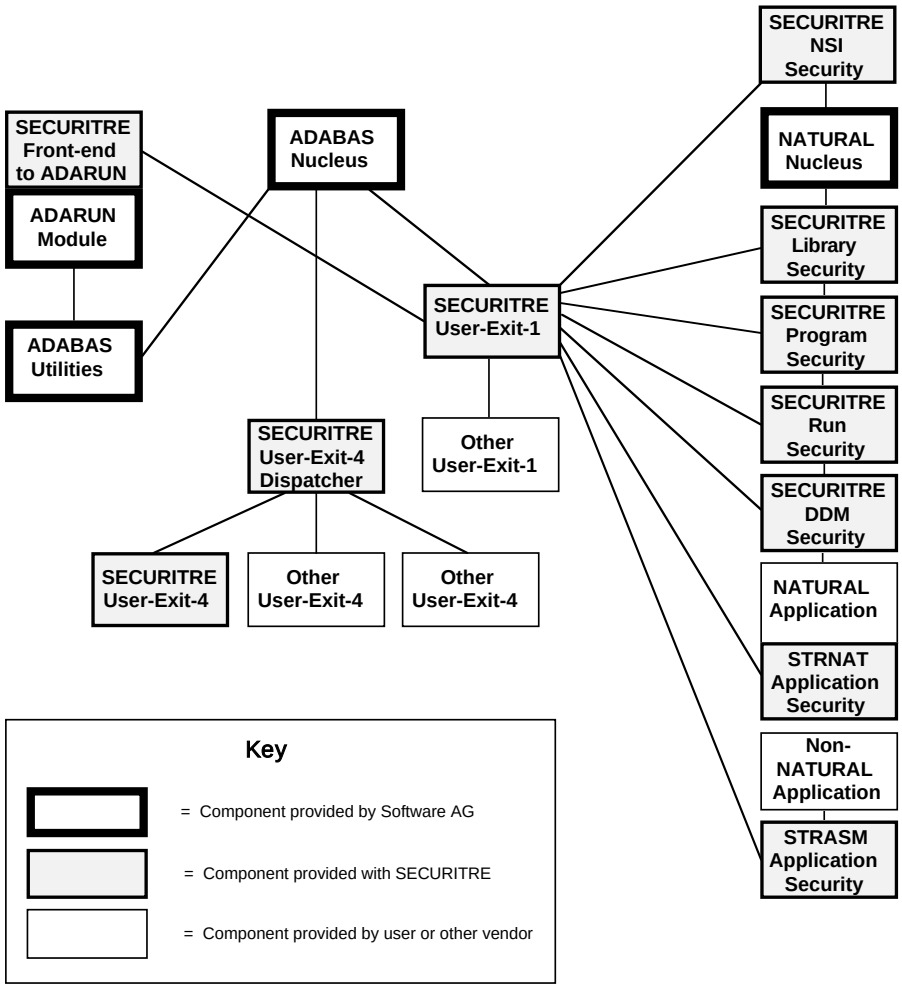

Figure 3 – SECURITRE MODULES

As shown above, the SECURITRE User-Exit-1 to the ADABAS nucleus is the "core" component of the product. The other components, such as the SECURITRE front-end to ADARUN and the SECURITRE for NATURAL options, rely on the SECURITRE User-Exit-1 for communication with the SSF. The optional components, such as the SECURITRE User-Exit-4 Dispatcher, need not be installed if they are not required, as is the case if the site uses no other User-Exit-4s.

The following sections describe how each SECURITRE component is installed, configured for use, and fine-tuned for use at the site.

This page intentionally left blank.

SECTION II

SETTING UP SECURITRE FOR ADABAS

II.1 Introduction

Within the ADABAS environment, the Security Administrator needs to control:

- Access to ADABAS databases
- Access to ADABAS files
- Access to specific fields contained in ADABAS files
- Methods used to access ADABAS resources, such as terminal, command, program, and job

SECURITRE offers these levels of control through a sophisticated User-Exit-1 to ADABAS.

II.2 User-Exit-1 to ADABAS

All calls received by the ADABAS nucleus are passed to the SECURITRE User-Exit-1 to ADABAS. SECURITRE User-Exit-1 is the "core" of the product. It is the most used and most often executed module of SECURITRE. When ADABAS calls are issued from any source, SECURITRE User-Exit-1 verifies that the user has the authority to access the requested ADABAS data. Security checks can not be avoided.

User-Exit-1 is primarily used for ADABAS file security and field level security. File security is always checked first. If file access is allowed and if SECURITRE Field Level Security is active for the file, SECURITRE authorizes the user to retrieve or update the field(s) stated in the ADABAS call.

SECURITRE "checks" ADABAS access by calling the SSF. Interface to the SSF is accomplished through the Security Authorization Facility (SAF) protocol, allowing SECURITRE calls to be consistent for all three SSFs. SSF conventions require standard OS datasets to be defined to the SSF as accessible or not accessible per User-ID.

The Security Administrator defines ADABAS databases, files, and fields to the SSF as pseudo OS datasets, such as "ADABAS.PROD.REVENUE," or "T43X7.ADA.P3.Ddddd.Ffffff", where 'ddddd' is the database number and 'ffffff' is the file number. The Security Administrator defines these pseudo-datasets, or DSNs, to SECURITRE User-Exit-1 through a set of flexible macro-generated parameters. The Security Administrator may define as many or as few DSNs as needed to identify the resources to be secured.

Subsequent attempts by the same User-ID to access the same DSN usually do not require redundant calls to the SSF because SECURITRE stores the results of previous calls in internal tables. These include a user table, a DSN table, user-to-DSN relationship table, and a field-security table. This enables SECURITRE to authorize "repeat" calls without contacting the SSF. Therefore, SECURITRE has minimal impact on ADABAS response time.

In addition, SECURITRE maintains information about each user's most recent accesses, enabling high-volume batch calls to be handled quickly and efficiently.

When a user issues a call to the ADABAS nucleus, the request is immediately passed through SECURITRE User-Exit-1, which is linked to the nucleus, before ADABAS processes it. SECURITRE examines and authorizes the call. This happens transparently to the user unless the user violates access rules. SECURITRE User-Exit-1 is in the ideal position to help the SSF control access to ADABAS resources without changing the way users access ADABAS.

II.3 Controlling Access to ADABAS Resources

The major SSFs only secure entire physical datasets. ADABAS data resides in two physical datasets, Data Storage (DATA) and Associator (ASSO). DATA and ASSO may contain data belonging to several logical databases, files, or fields. As a result, the SSF can only provide very limited security without SECURITRE in place. The SSF may either grant a user access to all of the data stored in the DATA and ASSO datasets or choose not to grant a user access to any of the data. For most sites, this control level is inadequate.

SECURITRE overcomes two major obstacles that enable the SSF to provide necessary control levels over access to ADABAS resources. These obstacles are:

- Identifying the user attempting to access ADABAS
- Dividing the ADABAS physical datasets by logical database, file, and field

Depending upon a site's security needs, SECURITRE controls ADABAS access at the database, file, and field levels. Security Administrators may combine these levels of control to achieve the necessary security for all ADABAS resources through the SSF.

II.3.1 Controlling Access on Database Level

With SECURITRE, the SSF may limit access to individual databases and files contained in DATA and ASSO. SECURITRE helps the SSF to identify the user and the database the user is attempting to access. This allows the site to control database-level access.

Controlling database-level access through SECURITRE is accomplished by assigning the same dataset name to all files belonging to the database. If a user is granted access to this dataset name, the user may access the entire database.

II.3.2 Controlling Access on File Level

Users must go through a link routine to access ADABAS files. The link routine communicates the user's access request to the ADABAS nucleus and returns the results of the request to the user. This communication travels from the user's region or address space into the region or address space occupied by the ADABAS nucleus and back.

From the ADABAS address space, it is not possible to determine the User-ID (SSF-ID) of the caller because the caller is in a different address space. The User-ID is necessary for SECURITRE User-Exit-1 to communicate with the SSF. Therefore, SECURITRE requires a User-Exit-B to be added to the link routine. This User-Exit-B operates in the user's address space, identifies the User-ID of the user, and includes the User-ID in the information communicated to ADABAS.

When ADABAS receives the call from the link routine, it passes the call to User-Exit-1. User-Exit-1 extracts the SSF-ID from the information passed by the link routine, identifies the database and file being accessed, and constructs the DSN for the call based on the STRPARMS coded by the Security Administrator. For example, the STRPARMS might indicate that the DSN for the Production Payroll file should be constructed as:

```
ADABAS.PROD.PAYROLL
```

SECURITRE then contacts the SSF to determine if the SSF-ID of the caller is authorized to access the "ADABAS.PROD.PAYROLL" dataset. The SSF checks its own rule base to determine if the request is authorized and returns a "yes" or "no" response to SECURITRE.

SECURITRE remembers the result of this access request in its tables and instructs ADABAS to process the request. ADABAS accesses the desired data and returns it to the link routine used by the caller. The link routine passes the requested data to the user.

When a call reaches the ADABAS nucleus and information is passed to SECURITRE User-Exit-1 for security checks, User-Exit-1 does not know the User-ID of the caller executing in another region/address space or on another CPU on the network. Users calling ADABAS in any environment, including batch, TSO, CICS, COM-PLETE, NATURAL, ADASQL, COBOL, and Direct Calls, must do so through an ADABAS link routine.

When a call is received by ADABAS User-Exit-1, SECURITRE determines if the command code is a type requiring a file number, such as a read, write, or find command. Other commands are ignored.

When SECURITRE User-Exit-1 to the ADABAS nucleus detects a read, write, or find command against the PAYROLL file in the PROD1 database, SECURITRE calls the SSF to determine if the user is authorized to access/update the file.

If file 78 on database 123 (PROD1) is the PAYROLL file, the Security Administrator may code the rule for file 78 in several forms. Examples include:

```
ADABAS.D123.F078
ADABAS.PROD1.F078
ADABAS.PROD1.PAYROLL
```

If either the database or the file number is greater than 255, the format of the rule that must be coded is different. For example, if file 1450 on database 1450 (DEV1) is the EMPLOYEES file, the Security Administrator may code the rule for file 1450 in one of the following ways:

```
ADABAS.D00200.F01450
ADABAS.DEV1.F01450
ADABAS.DEV1.EMPLOYEES
```

This change was necessary to accommodate the large database-IDs and file numbers that ADABAS Version 6 and above supports.

The Security Administrator may enter SSF rules to include or preclude access to the pseudo dataset for the PRODuction PAYROLL file to certain users or groups of users.

For efficiency, SECURITRE "remembers" authorization data, so that it does not have to call the SSF on every ADABAS call.

II.3.3 Controlling Access on Field Level

Field Level Security checks only go into effect after all of the following conditions are met:

- File Level Security has authorized the command
- Field Level Security is activated for the particular file
- Field(s) in the ADABAS Format Buffer match the field(s) being secured

Then, the field(s) being secured are attached to the DBID/FNR part of the rule or pseudo dataset name to form a new rule, such as:

```
ADABAS.PROD1.PAYROLL.SALARY
```

The Security Administrator may enter rules into the SSF to include or preclude access to the "pseudo dataset" for the SALARY field on the Production Payroll file to certain users or groups of users.

When SECURITRE User-Exit-1 to the ADABAS nucleus detects a read, write, or find command against the SALARY field in the PAYROLL file in the PROD1 database, SECURITRE calls the SSF to determine if the user is authorized to access/update the field(s).

For efficiency, SECURITRE "remembers" authorization data, so that it does not have to call the SSF on every ADABAS call.

II.4 Setting Up Security Parameters

This section describes the parameters used to customize SECURITRE for ADABAS.

II.4.1 Setting Up Database and File Security Parameters

When SECURITRE is initially installed, special attention should be given to certain parameters, while the fine tuning of other parameters may be saved for later. The parameters discussed in this section need immediate attention when the STRPARMS are first developed. The first parameters to be set up are SECURE, MODE, CLASS, DELIM, PREFIX, QUALIFY, USERID, USERID2, NOIDRED, and NOIDUPD. These parameters are defined in the STRDEF and STRFNR statements.

SECURE

Sites not using RACF, the default setting, need to change this parameter to "ACF2" or "TSS", since SECURITRE contains some RACF-specific code.

MODE

When first starting SECURITRE up, it is suggested that the MODE parameter be set to DORMANT mode, where User-Exit-1 will be invoked, but ADABAS commands will not be processed, or to WARN mode, where ADABAS commands will be processed, calls will be made to the SSF, any violations will be logged to the System Security Log, and SECURITRE will not prevent commands from being executed. The default, FAIL mode, will cause SECURITRE to return an ADABAS response code 200, "SECURITY VIOLATION," if the generated DSN is not authorized for the user. The STRDEF MODE parameter may be overridden at the file level in the STRFNR parameters, so while the STRDEF MODE may be DORMANT, certain files may be individually set to WARN or FAIL mode for security purposes.

The following chart shows the response code that will be returned by SECURITRE for ADABAS file security with various combinations of SSF and SECURITRE modes:

SSF Mode	SECURITRE Mode	SECURITRE Response Code
DORMANT	WARN	0
DORMANT	FAIL	0
DORMANT	DORMANT	0
WARN	WARN	0 (with warning message from the SSF)
WARN	FAIL	0 (with warning message from the SSF)
WARN	DORMANT	0 (with warning message from the SSF)
FAIL	WARN	0 (with warning message from SECURITRE)
FAIL	FAIL	200
FAIL	DORMANT	0

As shown in the table above, the only way SECURITRE will stop user operation is to have the SSF system and SECURITRE in FAIL mode. While SECURITRE is installed in a test environment, the site may wish to set SECURITRE in warn mode. However, the SSF should always be in FAIL mode to prevent unwanted access to resources.

Note: Some SSF products will lock-out a User-ID after a specified number of failed attempts. Since SECURITRE does not have control over this, the User-ID will be locked out even if SECURITRE is set to WARN or DORMANT mode.

CLASS
The default CLASS of DATASET is useful when SECURITRE is being tested and may also be useful at sites where SECURITRE will not generate a large number of different DSNs. Initially, it may be simplest to use the default.

When used in a production environment, CLASS=DATASET has limitations because the SSFs, most notably RACF, may take large amounts of memory for each user when generic security rules are used. Therefore, sites may decide to set up a special class for SECURITRE.

A drawback of using a class other than DATASET is that the SSF may not allow DSN masking for different classes. For example, a DSN rule may be set up for "ADABAS.PROD.F123.*" but not for "ADABAS.PROD.*.F123". Therefore, sites will need to carefully consider how CLASS is specified when implementing SECURITRE in a production environment. In addition, the DSNORDR parameter discussed later may impact how the user chooses to set the CLASS parameter.

DELIM, PREFIX, and QUALIFY
Both the DELIM and QUALIFY parameters may be set to null, specified as '' (two single quotes). This is useful if very short DSNs are required or if the QUALIFY parameter is regarded as unnecessary for a particular site's processing.

Note: Before deciding to use a null delimiter, it is important to consider that since the Utility and RTM Security functions also use DELIM, invalid DSNs consisting of more than eight characters may result, depending on the way UTORDER and RTMORDR have been specified.

DSNORDR
SECURITRE may be used to limit access to ADABAS files to particular jobs, programs, nodes, etc. through use of the DSNORDR parameter. In this way, SECURITRE may offer limited security on some files, while at the same time offering very tight security on other files in the same database. DSNORDR allows the user to specify that the DSNs include up to eight pieces of information about the way that the file is being accessed. The user defines the order in which the DSN will be generated.

USERID, USERID2
USERID2 is provided for special circumstances. It is not required for SECURITRE to look in two different places for a User-ID. It is simplest to have SECURITRE look in the USERINFO Area available in ADABAS.

To save a few Assembler commands, it helps to assign the most frequently used method to USERID and the less frequently used method to USERID2. For more information about these parameters, refer to the SECURITRE Reference Manual.

NOIDRED, NOIDUPD
There are valid instances where a User-ID will not be found. For example, an asynchronous task under CICS does not have a User-ID, but it is possible to derive a User-ID for SECURITRE to use. For example, the TRANID may be used. A SECURITRE User-Exit-1 (STREX1) may be developed to define what SECURITRE should do in these instances. During initial testing, an STREX1 user-exit may not be necessary. STREX1 is discussed in greater detail later in this section, and its calling parameters are listed in Section VII.2 - STREX1 User-Exit.

II.4.2 Setting Up Field Level Security

When SECURITRE is initially installed, the Security Administrator may not wish to activate Field Level Security. However, when the Security Administrator is ready to implement field level security for a file, the STRFNR parameters (FIELDS, FLSMODE, and FLSDEL) must be defined. Two of these parameters (FIELDS and FLSMODE) are of importance during the initial setup of Field Level Security.

FIELDS

For performance and efficiency, it is recommended that the number of fields to be secured be kept to a minimum. Remember that users cannot get to the fields unless they have authorization to access the file first.

FLSMODE

When first setting up Field Level Security, it is suggested that the FLSMODE parameter be set to DORMANT mode, where field level security is not checked or to WARN mode, where field level security is checked, calls will be made to the SSF, any violations will be logged on the System Security Log, and SECURITRE will not prevent commands from being executed. FAIL mode will cause SECURITRE to return an ADABAS response code 200, "SECURITY VIOLATION", if the field is not authorized for the user.

II.5 Using File Security Parameters to Control Access at the File and Database Levels

SECURITRE File Security Parameters can be used to control access at the database or file level. For example, consider the following parameter:

```
STRDEF  PREFIX=ADABAS,DELIM='.',QUALIFY='TEST',MODE=FAIL,
                DSNORDR=(FILE)
```

With these parameters in effect, an attempt to access file 123 on the Test database would cause SECURITRE to check the user's authority to access the DSN:

```
ADABAS.TEST.F123
```

If the file number is > 255, the value for the file is Fffffff, where fffff is the file number. An attempt to access file 272 would cause SECURITRE to check the user's authority to access the DSN:

```
ADABAS.TEST.F00272
```

Thus, access to each file is effectively controlled. If the site wished to break up the 255 files on the database into two groups, the following statements could be coded based upon the application which uses those files:

```
STRDEF  PREFIX=ADABAS,DELIM='.',QUALIFY='TEST',MODE=FAIL,
                DSNORDR=(FILE)
STRFNR  NAME=PAYROLL,FILE=(1-115)
STRFNR  NAME=BENEFIT,FILE=(116-300)
```

In this example, any attempt to access one of the Payroll files, such as 1, 2, 34, or 56, would generate the following DSN:

```
ADABAS.TEST.PAYROLL
```

Any attempt to access one of the Benefit files, such as 116, 180, or 280, would generate the following DSN:

```
ADABAS.TEST.BENEFIT
```

Thus, security for the 255 files on the TEST database does not have to be administered on a file-by-file basis. Instead, it can be controlled by a logical group of files, as specified in the STRFNR statement.

To control access at the database level, the site would code the following statements:

```
STRDEF  PREFIX=ADABAS,DELIM='.',QUALIFY='TEST',MODE=FAIL,
                DSNORDR=(FILE)
STRFNR  NAME=FILES,FILE=(1-255)
```

With these parameters in effect, an attempt to access any file on the Test Database would cause SECURITRE to check the user's authority to access the following DSN:

```
ADABAS.TEST.FILES
```

As a result, only one SSF rule needs to be maintained to control access to all of the 255 potential Test database files. SECURITRE is now effectively controlling ADABAS access at the database level.

II.6 Fine Tuning

After SECURITRE is up and running on a test system, more careful attention to certain SECURITRE parameters will be required. This sub-section highlights these parameters, which include USERS, USRPOOL, DSNPOOL, DSNORDR, and FSLPOOL, as well as the STRFNR parameters, which include DSNORDR, NAME, FIELDS, and FLSMODE.

User-Exit-1 Table Size Parameters

The USERS, USRPOOL, DSNPOOL, and FLSPOOL parameters specify how much space to allocate for the SECURITRE tables. The following figure shows the relationships between these three tables.

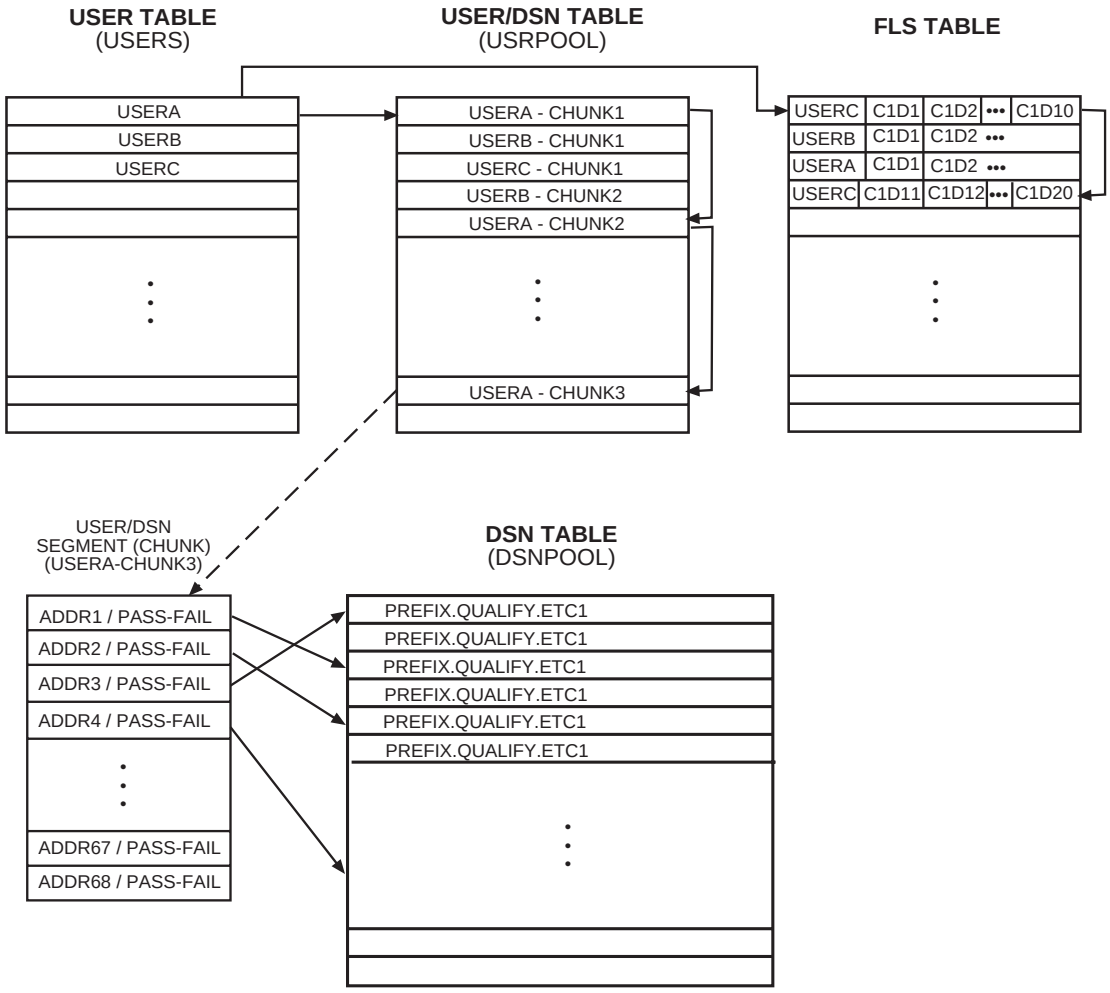

Figure 4 – Relationships Between User-Exit-1 Tables

USERS
For maximum efficiency, it is best to set this parameter to a number that is slightly higher than the estimated number of users at peak usage time. If USERS is set too low, SECURITRE will need to spend more time purging users from the table as it fills up. If USERS is set too high, allocated storage will be wasted.

USRPOOL
Each segment of the USRPOOL contains 68 entries, and each entry contains a pointer to a DSN in the DSN table plus flags indicating what kind of access (READ/WRITE) was attempted for the DSN and whether the access was successful. Each segment may belong to only one user, but one user may have many segments.

An initial value for USRPOOL may be calculated as:

USRPOOL = USERS * (avg #DSNs per user / 68)

In this case, (avg. #DSNs per user / 68) is rounded up. The value assigned to USRPOOL **must** be divisible by four.

If all the USRPOOL segments are being used and SECURITRE needs to allocate a new segment, the oldest users as defined by the PURINTT parameter, and their segments will be purged from the tables. When this happens, SECURITRE will produce a message on the operator's console and in the STRMSG dataset indicating "CHUNK POOL FULL - INCREASE SIZE."

The value for the USRPOOL parameter should be increased if SECURITRE produces "CHUNK POOL FULL" messages on a regular basis.

DSNPOOL
When assigning a value to the DSNPOOL parameter, consider the following:

- The value of the DSNORDR parameter will affect the number of DSNs that can be generated.

- The DSNs generated by SECURITRE for NATURAL are maintained in the DSN table in User-Exit-1. If Program Security is used, many DSNs may be generated. This may make it necessary to allocate a large DSNPOOL. However, maintaining the SECURITRE for NATURAL DSNs in this table may substantially reduce the number of calls made to the SSF.

- The DSN table maintains DSNs for ALL users. Therefore, if the same DSN is generated for ten different users, the same entry in the DSN table will be referenced for all ten users.

- When the DSN table is full, SECURITRE purges all DSNs from the DSN table and all user entries from the user table. Therefore, it is a good idea to set a high value on the DSNPOOL parameter in order to avoid regenerating the tables.

DSNORDR

What should be included in the DSN depends on the site's environment and security needs.

DSNORDR can only be used if SECURITRE obtains the User-ID using method STRUEXB or ALT-2. DSNORDR obtains information from the SECURITRE USERINFO area, which is created from the SECURITRE User-Exit-B (STRUEXB, STRUEXBB, STRUEXB2, STRUEXB3, and STRUEXB5).

SECURITRE will generate the DSN beginning with the PREFIX and QUALIFY parameters and then add items to the DSN according to the order specified in the DSNORDR parameter. It will not include items that are meaningless in the context of the call. For example, it will not try to include a CICS Transaction-ID if the call does not originate from CICS. DSNs generated by SECURITRE are limited to 44 characters. When SECURITRE determines that adding an item to the DSN exceeds this limit, it will not include any of the remaining items.

The DSNORDR parameter may be overridden at the file level in the STRFNR parameters. Therefore, it is possible to set up some files for very strict security requirements while leaving other files less stringently secured. For more information about the DSNORDR items, refer to the SECURITRE Reference Manual.

The DSNORDR parameter should be set up for a file depending on how secure the file should be. For less sensitive files that are accessed by many people, but are too sensitive for DORMANT mode, it is sufficient to set DSNORDR=FILE. Files that are very sensitive may need more items in the DSNORDR parameter.

DSNORDR Examples

For the following DSNORDR parameter examples, assume that PREFIX='ADA' and QUALIFY='' and the values below have been obtained by SECURITRE for the different DSNORDR items:

```
FILE=SALARY
JOB=CICSPROD, TSOUSER or BATCHUSR
CMD=S1 (find) or N1 (add)
NODE=N0002
DBID=D123F101
NPGM=NATPAYPG
NLIB=PAYLIB
GPGM=ANYPROG or GENPAYPG
CICSTRAN=NAT2
CICSTERM=T123
```

Example 1:

The SALARY file may only be accessed by the Production CICS NATURAL environment running on node 0002 and from programs in the library PAYLIB. The DSNORDR parameter for this file has been set to:

```
DSNORDR=(NODE,JOB,NLIB,FILE)
```

If a user is accessing the file properly, the following DSN will be generated:

```
ADA.N0002.CICSPROD.PAYLIB.SALARY
```

SECURITRE will check with the SSF to see if the user has access to this DSN. If the same user has written a COBOL program to look at the SALARY file then the following DSN will be generated and access will be denied.

```
ADA.N0002.TSOUSER.SALARY
```

Example 2:
The same rules used in Example 1 apply, except that only specific terminals may be used to access the SALARY file. The DSNORDR parameter for file 123, the SALARY file, is set to:

```
DSNORDR=(JOB,NLIB,NPGM,GPGM,TERM,FILE)
```

When a user that is authorized to access the SALARY file attempts to access this file at that user's own terminal, SECURITRE will generate the following DSN:

```
ADA.CICSPROD.PAYLIB.NATPAYPG.T123.SALARY
```

When users discover their passwords and try to update the file on a different terminal, T059, SECURITRE will generate the following DSN and access will be denied by the SSF:

```
ADA.CICSPROD.PAYLIB.NATPAYPG.T059.SALARY
```

Terminal Security is not applicable in batch. When paychecks are written from a batch COBOL program, SECURITRE will generate the following DSN:

```
ADA.BATCHJOB.GENPAYPG.SALARY
```

Example 3:
The ADAULD utility uses the ADABAS commands LF and LA. A particular employee is allowed to run unloads against the SALARY file, but otherwise is not allowed access to that file. In this case, the site may want to set DSNORDR=(FILE,CMD) so this user can only perform LF and LA commands against the file. When the user attempts the ADAULD, SECURITRE will generate the following DSNs:

```
ADA.SALARY.LF
ADA.SALARY.LA
```

However, when the user attempts run any other commands against the file, such as S1 (FIND), L3 (READ), or A1 (UPDATE), access will be denied by the SSF.

Hint: The TRACE facility may be turned on for short periods of time on a low-usage database in order to determine which commands are generated by a program.

Example 4:
A site is using the same database for both Test and Production FUSERs. The Production FUSER is D123F101 and the Test FUSER is D00123.F00256. A user has access to update the production SALARY file from the PAYLIB library. However, this user cannot create or change programs, and the programs in PAYLIB are only used to update non-sensitive fields in the SALARY file. The user, who also has access to PAYLIB in the Test system, writes a program to activate a raise for that user. The DSNORDR parameter for file 123, the SALARY file, is set to:

```
DSNORDR=(DBID,NLIB,FILE)
```

SECURITRE generates and checks the following DSN and access to the file is denied by the SSF:

```
ADA.D00123.F00256.PAYLIB.SALARY
```

Example 5:
A clerk in the Payroll Department is allowed to see and update addresses but is not allowed to look at salaries. The clerk runs a program that is used to display salary information obtained from the PAYROLL file fields EE and EF. The DSNORDR parameter for file 210, the PAYROLL file, is set to:

```
DSNORDR=(FILE,FIELD)
```

The FIELDS parameter is set to:

```
FIELDS=(EE,SALARY,EF,SALARY).
```

SECURITRE will generate and check the following DSNs:

```
ADA.PAYROLL
ADA.PAYROLL.SALARY
```

Since the clerk does not have access to ADA.PAYROLL.SALARY a response code 200 will be received, and the information will not be displayed.

Note: If both the Database-ID and the file number are < 256, DBID is formatted as DxxxFyyy where xxx is the Database-ID, and yyy is the file number. If either the Database-ID or the file number is > 255, DBID is formatted as DxxxxxFyyyyy, where xxxxx is the Database-ID, and yyyyy is the file number.

II.7 Cleaning Up: PURINTT, PURINTV, FORCE, PROCCL

The user table is maintained in an order sorted by User-ID, so that a binary search may be performed when SECURITRE needs to obtain a user's entry – the fewer number of entries in the table, the quicker the search. SECURITRE "cleans up" its tables according to the values specified in the PURINTT, PURINTV, and FORCE parameters defined in the STRDEF statement in order to keep the number of table entries to a minimum.

The PURINTT parameter specifies how many seconds must pass after a user issues the last command before they are considered an "inactive" user. SECURITRE will only reference the PURINTT parameter when it is in the process of purging the table. This may be a scheduled purge (refer to PURINTV and FORCE below), or it may occur when the user table is full. If the table is being purged hourly, it is possible for the user to be inactive for up to 59 minutes and 59 seconds and still not be considered inactive at the time of the purge.

The PURINTV parameter specifies how many hours must pass before SECURITRE will purge the user table of inactive users. When a user entry is deleted, the corresponding entries in the User-to-DSN relationship table are also deleted.

The FORCE parameter specifies what time of day SECURITRE should purge all entries from the user table. A site may prevent a daily purge from taking place by setting FORCE=99.

When a user issues a CL (CLOSE) command, that user's User-ID remains in the table, but all information related to the User-ID, including the user's ACEE, is deleted. SECURITRE keeps the User-ID in the table to prevent moving large amounts of data while keeping the table in sorted order. The User-ID entry is deleted when a purge takes place. When the user's ACEE is deleted, SECURITRE must call the SSF with a RACINIT CREATE before it can make any further requests of the SSF for that user.

Normally, this is desirable because SECURITRE will be re-synchronized with the user's security rules in the SSF when the RACINIT takes place. However, this causes a lot of extra work for sites that issue multiple CL commands and for sites with ADASQL which issues frequent CL commands. Sites, as described above, should set the PROCCL parameter OFF. All other sites should use the default PROCCL value of ON.

For a database with a large number of potential users who use the database infrequently during the day, a frequent purge may be desirable, since reducing the number of users in the table will decrease search time.

For databases where users are constantly accessing the files or where there are a small number of users, fewer purges of the table are necessary.

II.8 Special Needs: STREX1, STREX2, and NOIDRED/NOIDUPD

To determine the User-ID of the user accessing the database, SECURITRE depends on the settings of the USERID and USERID2 parameters, as well as the proper installation of either the SECURITRE User-Exit-B for ADABAS.

STREX1

In some instances SECURITRE will not be able to obtain a User-ID, either for valid reasons, such as an asynchronous CICS transaction, or due to errors in the installation process. To handle such situations, SECURITRE includes its own User-Exit-1, which may be coded at the site. A sample SECURITRE User-Exit-1 is included on the tape. For more information, refer to the Installation section of this manual.

If SECURITRE does not find a User-ID using the methods assigned to USERID or USERID2, it will call STREX1 if it exists. By examining data in the USERINFO Area, users may be able to either assign a User-ID or tell SECURITRE whether to ACCEPT or REJECT the command. For more information, refer to the coding STREX1 in the User-Exits to SECURITRE section of the SECURITRE Reference Manual.

NOIDRED/NOIDUPD

If there is still no value assigned to the User-ID, SECURITRE will then look at the values assigned to the NOIDRED and NOIDUPD parameters. If the appropriate parameter is set to ACCEPT, SECURITRE will allow the command without further security checking. If the appropriate parameter has been set to REJECT, SECURITRE will issue an ADABAS response code 200, "SECURITY VIOLATION".

STREX2

Sites that use ADABAS password security, in which a password is included in Additions-3 of the ADABAS Control Block, may use the SECURITRE Exit 2, STREX2, to obtain the password from the SSF and assign it to Additions-3. In this manner, passwords may be maintained in the SSF, and users will not need to provide them. With this mechanism, the site does not need to be concerned about the sharing of passwords among users. Using STREX2 is especially useful for ADABAS Security By Value.

SECURITRE calls STREX2 (if it exists) after processing the command for file and field security, but before the command is passed to ADABAS. After a password has been assigned to Additions-3, it is copied to the user's entry in the User Table, so that it may be assigned again on subsequent calls without calling STREX2.

A sample STREX2 that assigns the USERID into Additions-3 is provided on the tape.

II.9 Securing the RTM

The STRRTM and RTMORDR parameters are used to control access to the SECURITRE RTM. These parameters are defined in the STRDEF statement.

STRRTM

STRRTM designates the prefix used when generating DSNs for RTM security. The STRDEF/STRFNR PREFIX and QUALIFY parameters are **not** used when generating DSNs for RTM Security.

RTMORDR

The valid values are DBID and FUNC. The DBID will appear in the DSN in the form 'Dnnn', where 'nnn' is the number of the database, such as D073 in the examples below. If the DBID is greater than 255, the DBID will appear in the DSN in the form 'Dnnnnn', where 'nnnnn' is the number or the database, such as D00300 in the examples below.

Example 1:

If STRRTM='ADABAS.RTM' and RTMORDR=DBID, when a user accesses the RTM, SECURITRE will generate the following DSN:

```
ADABAS.RTM.D073 or ADABAS.RTM.D00300
```

Example 2:

If STRRTM='ADABAS.RTM' and RTMORDR=(FUNC,DBID), when a user attempts to load new SECURITRE parameters, SECURITRE will generate the following DSN:

```
ADABAS.RTM.LODPARM.D073 or ADABAS.RTM.LODPARM.D00300
```

Example 3:

If STRRTM='ADABAS' and RTMORDR=(DBID,FUNC), when a user attempts to load new SECURITRE parameters, SECURITRE will generate the following DSN:

```
ADABAS.D073.LODPARM or ADABAS.D00300.LODPARM
```

Note: The MODE for the RTM is always FAIL. RTM Security requires READ access. When installing SECURITRE and attempting to use the RTM for the first time, DBAs or Security Administrators can 'pre-approve' themselves by setting the STRRTM parameter to a high-level qualifier they are already allowed to access.

When "FUNC" is specified in the RTMORDR parameter, the following "Function in DSNs" is used when generating the DSNs:

SCREEN NAME	SCREEN FUNCTION	FUNCTION in DSN
FRC1	Force One User From the Tables	FRC1USR
FRCA	Force All Users From the Tables	FRCAUSR
PARM	Display SECURITRE Parms	MONSTAT
REXT	Reload User-Exit(s)	LODPGM
RPRM	Reload SECURITRE Parms	LODPARM
TRAC	SECURITRE Trace Facility	TRACON TRACOFF
NPRM	Display SECURITRE/NATURAL Parms	STRNPRM PARMUPD
TBLS	Display SECURITRE Table Sizes	TABSZ TRACOFF

Figure 5 – RTM Security Suffixes

Note: The "Function in DSN" column lists the values the Security Administrator should use for "FUNC" when defining the dataset name.

II.10 Interfaces to Other Products

Some SECURITRE components are built-in to allow SECURITRE to co-exist with other products. The following is a discussion of the SECURITRE tools to use when combining it with other products.

UEXIT1

The UEXIT1 parameter defined in the STRDEF statement specifies the name of an ADABAS User-Exit-1 module to be given control after SECURITRE has completed its processing. For example, if SECURITRE is running with TRIM, the UEXIT1 parameter should be set to TRMUEX1.

Dispatcher for User-Exit-4

If multiple ADABAS User-Exit-4s are necessary, SECURITRE includes a dispatcher that should be installed. For more information about the operation of the User-Exit-4 dispatcher, refer to **Section VIII.2 - User-Exit-4 Co-Existence**.

II.11 STRDEF Statements Sample

The STRDEF sample statement below covers most of the parameters discussed in the preceding subsections.

Only one STRDEF statement may be coded, but the statement may be as long as needed to accommodate all the desired parameters.

Col. 72

```
*
*     SET THE DEFAULT VALUES FOR ALL FILES IN THIS DATABASE
*
       STRDEF CLASS=ADABAS,PRINT=NOGEN,USERS=75,                          x
              CMDLOG=ON,FLSDEL=DELETE,FLSPOOL=10,PROCCL=ON,               x
              DELIM='.',PREFIX='SECURED',QUALIFY='PAYROLL',                x
              DSNORDR=(JOB,FILE),DSNPOOL=100,USRPOOL=200,                 x
              FORCE=09,LOGVIOL=FIRST,MODE=FAIL,TERM=T,                    x
              NOIDRED=ACCEPT,NOIDUPD=REJECT,                              x
              PURINTT=300,PURINTV=1,                                      x
              RACHECK=RACHECK,SECURE=RACF,                                x
              STREX1=NOUSER,UEXIT1=TRMUEX1,                               x
              USERID=STRUEXB,USERID2=TRIMV4-2,                            x
              RTMORDR=(DBID,FUNC),STRRTM='CONTROL.STR',                   x
              TRMRTM='CONTROL.TRIM',                                      x
              UTORDER=(UTIL,FILE,FUNC),UTPREF='ADAUTIL'
```

An explanation of these sample parameters follows:

- **CLASS=ADABAS**
  The resource class is "ADABAS".

- **PRINT=NOGEN**
  SECURITRE will inform the assembler not to cause macro expansions to be printed in the assembler listing.

- **USERS=75**
  SECURITRE will maintain up to 75 users' entries in the SECURITRE internal tables.

- **CMDLOG=ON**
  SECURITRE will request that all commands be logged by ADABAS. This may be overridden by the execution of subsequent User-Exit-4s.

- **FLSDEL=DELETE**
  When a user is deleting a record in a file where FLSMODE is WARN or FAIL, SECURITRE will use DELETE in the DSN in place of a field alias.

- **FLSPOOL=10**
  SECURITRE will maintain Field Level Security information for 10 ADABAS Command-IDs per user at any given time.

- **PROCCL=ON**
  Any user issuing an ADABAS CL command will be removed from the SECURITRE tables.

- **DELIM='.'**
  SECURITRE will use a period (".") as the delimiter when generating the SSF DSN for authorization checks.

- **PREFIX='SECURED',UTPREF='ADAUTIL'**
  SECURITRE will prefix each DSN with "SECURED", except for utility runs. They should be prefixed with "ADAUTIL".

(continued on next page)

(continued from previous page)

- **QUALIFY='PAYROLL'**
  SECURITRE will use the default qualifier "PAYROLL".

- **DSNORDR=(JOB,FILE)**
  SECURITRE will include, in addition to the prefix and qualifier, the MVS jobname and the file alias in the DSN.

- **DSNPOOL=100**
  SECURITRE will handle up to 100 DSN table entries.

- **USRPOOL=200**
  SECURITRE will handle up to 200 user-to-DSN relationship table entries.

- **FORCE=09**
  SECURITRE will purge the internal table of all users at 09:00.

- **LOGVIOL=FIRST**
  SECURITRE will cause the SSF to log only the first violation for a given user to a given file.

- **MODE=FAIL**
  SECURITRE will not allow any command not authorized by the SSF to be processed by ADABAS (i.e., FAILed).

- **TERM=T**
  The SECURITRE RTM NATURAL programs will TERMINATE rather than STOP (i.e., the user will be taken out of NATURAL when they exit the RTM).

- **NOIDRED=ACCEPT**
  SECURITRE will allow READ commands to process when the User-ID is unknown.

- **NOIDUPD=REJECT**
  SECURITRE will not allow UPDATE commands to process when the User-ID is unknown.

- **PURINTV=1,PURINTT=300**
  SECURITRE will purge the SECURITRE internal tables of all entries that have been inactive for 5 minutes (300 seconds) or more every hour.

- **RACHECK=RACHECK**
  SECURITRE will use the normal "RACHECK" method for authorizing access.

- **SECURE=RACF**
  The System Security Facility (SSF) used by this site is RACF.

- **STREX1=NOUSER**
  SECURITRE will invoke user-exit module NOUSER if it cannot determine the User-ID that issued a given command.

- **UEXIT1=TRMUEX1**
  SECURITRE User-Exit-1 to ADABAS will invoke program TRMUEX1 after it has completed processing.

(continued on next page)

(continued from previous page)

- **USERID=STRUEXB**
  The method SECURITRE will first use to attempt to locate the SSF User-ID is by examining the SECURITRE USERINFO Area.

- **USERID2=TRIMV4-2**
  If the first method for locating a User-ID fails (USERID), SECURITRE will attempt to find the SSF User-ID by examining the Additions-4 field.

- **RTMORDR=(DBID,FUNC),STRRTM='CONTROL.STR'**
  Any user attempting to execute the SECURITRE RTM must have access to the "CONTROL.STR.Dnnn.RTM-function" or "CONTROL.STR.
  Dnnnnn.RTM-function" DSN.

- **TRMRTM='CONTROL.TRIM'**
  Any user attempting to execute the TRIM RTM must have access to the "CONTROL.TRIM" DSN.

- **UTORDER=(UTIL,FILE,FUNC),UTPREF='ADAUTIL'**
  SECURITRE will generate DSNs for Utility Security as "ADAUTIL.utility.file.function".

Note:	A complete explanation of each of these parameters is presented in the SECURITRE Reference Manual.

II.12 STRDEF/STRFNR Statements Sample

The sample statements below will set SECURITRE to function in an ADABAS environment.

Col. 72

```
*
*        SET THE DEFAULT VALUES FOR ALL FILES IN THIS DATABASE
*
         STRDEF CLASS=DATASET,            DEFAULT                              x
                DELIM='.',                SEPARATE DSN PARTS WITH PERIOD       x
                DSNORDR=(FILE,FIELD),     INCLUDE FILE/FIELD NAME IN DSN       x
                DSNPOOL=100,              100 DSNs IN TABLE                    x
                FILEMAX=NEW               NEW FORMAT FILE PARMS                x
                FLSDEL=DELETE,            LITERAL FOR FLS ON DELETE CMDS       x
                FLSPOOL=10,               RETAIN 10 CIDS FOR FLS PROCESSING    x
                FORCE=2,                  PURGE ALL USERS AT 2 A.M.            x
                LOGVIOL=ALL,              LOG ALL VIOLATIONS                   x
                MODE=FAIL,                FULL IMPLEMENTATION STATUS           x
                NOIDRED=REJECT,           NO USERID READ - REJECT              x
                NOIDUPD=REJECT,           NO USERID UPDATE - REJECT            x
                PREFIX='ADABAS.SECTRE',   FIRST PART OF DSN                    x
                PRINT=NOGEN,              SAVE PAPER                           x
                PURINTV=2,                PURGE INACTIVE USERS EVERY 2 HRS.    x
                PURINTT=1800,             INACTIVITY IS 30 MINUTES             x
                QUALIFY='PROD',           SECOND PART OF DSN                   x
                RACHECK=RACHECK,          DEFAULT                              x
                RTMORDR=FUNC,             INCLUDE RTM FUNCTION IN DSN          x
                SECURE=ACF2,              ACF2 INSTALLED                       x
                STREX1=NOUSER,            EXIT ON NO USERID                    x
                STRRTM='CONTROL.STR',     DSN TO PROTECT STR RTM               x
                TERM=S,                   STOP RTM ON EXIT                     x
                TRACE=OFF,                PRODUCE NO TRACE MESSAGES            x
                TRMRTM='CONTROL.TRM',     DSN TO PROTECT TRM RTM               x
                UEXIT1=TRMUEX1,           INVOKE TRIM UEX1 ALSO                x
                USERID=STRUEXB,            USERID IN USERINFO AREA             x
                USERID2=TRIMV4-1,         USERID IN ADDITIONS-3                x
                USERS=200,                HOLD 200 USERS IN TABLE              x
                USRPOOL=400,              400 USER/DSN SEGMENTS IN TABLE       x
                UTMODE=FAIL,              ABEND UNAUTHORIZED UTILITIES         x
                UTORDER=UTIL,             INCLUDE ONLY UTILITY IN DSN          x
                UTPREF='ADAUTIL'          PREFIX FOR UTILITIES
*
*        USE A DIFFERENT NAME FOR PAYROLL FILE AND INCLUDE THE JOB,
*        NATURAL LIBRARY (IF AVAILABLE), THE PROGRAM NAME AND THE FILE
*        NUMBER IN THE DATASET NAME
*
           STRFNR FILE=001,                                                   x
                PREFIX='PAYROLL',                                             x
                QUALIFY='EMPLOYEE',                                           x
                NAME='PAYRATE.F01',                                           x
                DSNORDR=(JOB,NLIB,NPGM,GPGM,FILE)
*
*        ALLOW READ ACCESS EVEN IF USERID IS UNKNOWN
*
         STRFNR FILE=010,                                                     x
                PREFIX='PUBLIC.ACCESS',                                       x
                QUALIFY='OPEN',                                               x
                NAME='VENDOR.F010',                                           x
                NOIDRED=ACCEPT,                                               x
                NOIDUPD=REJECT
*
*        NEW FILE IS WARN MODE FOR AWHILE
*
         STRFNR FILE=011,                                                     x
                NAME='NEW.FILE.F011',                                         x
                MODE=WARN
*
*        EVERYONE HAS ACCESS TO THE PREFIX 'PUBLIC.ACCESS' SO
*        THIS FILE IS TOTALLY OPEN TO EVERYONE
*
```

Figure 6 – STRDEF/STRFNR Statements Sample

(continued on next page)

(continued from previous page)

Col. 72

```
STRFNR FILE=012,                                                        x
               PREFIX='PUBLIC.ACCESS',                                  x
               QUALIFY='OPEN',                                          x
               NAME='SPOOL.F012',                                       x
               MODE=WARN

*
*        A VERY SMALL ENTITY NAME IS DESIRED.  THE DSN GENERATED
*        WILL BE:  SMALF013
*
         STRFNR FILE=013,                                               x
                PREFIX='SM',                                            x
                QUALIFY='AL',                                           x
                DELIM=''
*
*        THESE FILES ARE IN WARN MODE
*
         STRFNR FILE=060-066,                                           x
                MODE=WARN
*
*        FIELD LEVEL SECURITY IS IN EFFECT FOR SOME FIELDS
*        ON THIS FILE
*
         STRFNR FILE=215,FLSMODE=FAIL,                                  x
                FIELDS=(FF,SALARY,GH,SALARY,XX,BONUS)
*
*        THESE FILES CAN BE READ IF NO USER ID IS FOUND FOR THE USER
*
         STRFNR FILE=300-900,                                           x
                NOIDRED=ACCEPT
         STRFNR END
         END
```

Figure 6 – **STRDEF/STRFNR Statements Sample**

Note:	Since FILEMAX=NEW is used in this example, STRFNR END is needed.

II.13 STRDEF and STRFNR Examples

This section provides some examples of complete STRDEF/STRFNR groups. Each example includes an explanation of the SECURITRE for ADABAS parameters used.

Example 1:
Assume the following parameters:

Col. 72

```
STRDEF      CLASS=ADABAS,DELIM='.',DSNORDR=(JOB,FILE),                    x
            DSNPOOL=200,FILEMAX=NEW,FLSPOOL=0,FORCE=2,                    x
            LOGVIOL=FIRST,MODE=DORMANT,NOIDRED=ACCEPT,                    x
            NOIDUPD=ACCEPT,PREFIX='ADABAS',PURINTT=600,                   x
            PURINTV=2,QUALIFY='PROD',RTMORDR=FUNC,SECURE=TSS,             x
            STRRTM='ADABAS.STR',TRMRTM='ADABAS.TRM',                      x
            UEXIT1=TRMUEX1,USERID=STRUEXB,USERID2=TRIMV4-2,               x
            USERS=100,UTMODE=FAIL,UTORDER=(UTIL,FILE),                    x
            UTPREF='ADABAS.UTL'                                           x
*
STRFNR      FILE=(1,5-25),NAME='PERSONEL',MODE=FAIL,                      x
            NOIDRED=ACCEPT,NOIDUPD=REJECT
*
STRFNR      FILE=100,NAME='PAYROLL',LOGVIOL=ALL,MODE=FAIL,                x
            NOIDRED=REJECT,NOIDUPD=REJECT
*
STRFNR      FILE=101,NAME='SALARY',MODE=FAIL,LOGVIOL=ALL,                 x
            NOIDRED=REJECT,NOIDUPD=REJECT,                                x
            DSNORDR=(JOB,NLIB,NPGM,GPGM,FILE)
*
STRFNR      FILE=(150-155),NAME='PRODFILE',MODE=FAIL,DSNORDR=JOB
*
STRFNR      FILE=900,NAME='NEWFILE',MODE=WARN
*
STRFNR      END
END
```

The following STRDEF parameters were not included, so SECURITRE will assign the default values indicated:

```
CMDLOG=OFF
FLSDEL=DELETE
PRINT=NOGEN
PROCCL=ON
STREX1=                (none)
STREX2=                (none)
TERM=S                 (stop)
TRACE=OFF
```

The STRDEF statements indicate the following for all files unless a STRFNR override statement is specified:

- **CLASS=ADABAS**
  SECURITRE will use the class "ADABAS" when making calls to the SSF.

- **DELIM='.'**
  SECURITRE will use a period as the delimiter character between all DSN components when DSNs are generated for File Security, Utility Security, and RTM Security.

(continued on next page)

(continued from previous page)

- **DSNORDR=(JOB, FILE)**
  For File Security on any file except files 101 and 150-155, SECURITRE will generate the DSNs as follows:

  ```
  ADABAS.PROD.jjjjjjjj.ffffffff
  ```

  In this case, 'jjjjjjjj' is the MVS jobname and 'ffffffff' is the file number (Fnnn) or (Fnnnnn), otherwise it is STRFNR file alias NAME override. For example, if a user is accessing file 001 from CICS production, SECURITRE will generate and check for authorization to the following DSN:

  ```
  ADABAS.PROD.CICSPROD.PERSONEL
  ```

  If a user is accessing file 002 from the TSO job 'JOEBATCH', SECURITRE will generate and check the following DSN:

  ```
  ADABAS.PROD.JOEBATCH.F002
  ```

  Note: If the file number is greater than 255, the value given to the file number will be formatted as 'Fnnnnn', where 'nnnnn' is the file number (e.g., F01234).

- **DSNPOOL=200**
  SECURITRE will allocate space for 200 DSNs in its DSNPOOL table.

- **FLSPOOL=0**
  SECURITRE will not allocate any space for Field Level Security processing.

- **FILEMAX=NEW**
  SECURITRE will generate a variable number of 80-byte file entries dependent on the number of files with STRFILE parameters.

- **FORCE=2**
  SECURITRE will purge its tables daily at 2 a.m.

- **LOGVIOL=FIRST**
  SECURITRE will cause the SSF to log only the first violation in the system security log for each user denied access to any file except files 100 and 101.

- **MODE=DORMANT**
  All files not mentioned in the STRFNR statements remain unsecured.

- **NOIDRED=ACCEPT**
  If SECURITRE is processing a READ command, such as L3 or S1, against any file except files 100 and 101, and it cannot obtain a User-Id, SECURITRE will still allow the command to be processed.

- **NOIDUPD=ACCEPT**
  If SECURITRE is processing an UPDATE command, such as A1 or E1, against any file except files 1, 5-25, 100, or 101, and it cannot obtain a User-ID, SECURITRE will still allow the command to be processed.

- **PREFIX='ADABAS'**
  SECURITRE will use the prefix "ADABAS" for creating the first part of the DSN when making calls to the SSF.

(continued on next page)

(continued from previous page)

- **PURINTT=600**
  Users who have not issued a command for more than 10 minutes (600 seconds) will be considered "inactive" by SECURITRE when a purge takes place.

- **PURINTV=2**
  SECURITRE will purge its tables of inactive users every 2 hours.

- **QUALIFY=PROD**
  SECURITRE will use the qualifier "PROD" for generating the second part of the DSN when making calls to the SSF.

- **RTMORDR=FUNC,STRRTM='ADABAS.STR'**
  When a user executes a secured RTM function, SECURITRE will generate the DSN as follows:

  ```
  ADABAS.STR.func
  ```

  In this case, 'func' is an abbreviated function utility. A complete list of utility functions used by SECURITRE can be found in ***Section IV.2 – ADABAS V5, V6, and V7 Utility Control*** in the ***SECURITRE Reference Manual***. For example, when a user runs the RTM function FRCA, SECURITRE will generate the following DSN:

  ```
  ADABAS.STR.FRCAUSR
  ```

- **SECURE=TSS**
  The SSF in use at this site is TOP SECRET.

- **TRMRTM='ADABAS.TRM'**
  When TRIM sends calls to SECURITRE to verify TRIM Security, the DSNs generated will begin with 'ADABAS.TRM'.

- **UEXIT1=TRMUEX1**
  When SECURITRE has finished its User-Exit-1 processing, it will pass control to TRMUEX1.

- **USERID=STRUEXB**
  SECURITRE will look for a USERINFO Area that it has generated in order to find the User-ID and other information.

- **USERID2=TRIMV4-2**
  This site also runs COM-PLETE, so it is necessary for SECURITRE to get the User-ID from Additions-4 for the calls coming from COM-PLETE.

- **USERS=100**
  This site expects about 90 users to use the database during peak periods. They are requesting that SECURITRE allocate space for 100 users. By padding this parameter, they will avoid the possibility of running out of space.

- **UTMODE=FAIL**
  SECURITRE Utility Security is in FAIL mode. The utility will abend and violations will be logged if the user is not authorized to access the utility.

(continued on next page)

(continued from previous page)

- **UTORDER=(UTIL,FILE),UTPREF='ADABAS.UTL'**
  SECURITRE Utility Security will generate the DSNs as follows:

  ```
  ADABAS.UTL.uuu.ffffffff
  ```

  In this case, 'uuu' refers to the last three characters of the utility name and 'ffff' refers to either the file number Ffffff or Ffff or its STRFNR alias NAME override. For example, when a user runs the ADAULD utility for file 900, Utility Security will generate and check authorization for the following DSN:

  ```
  ADABAS.UTL.ULD.NEWFILE
  ```

The STRFNR statement for **FILE=(1,5-25)** indicates that the following parameters will override the previously defined STRDEF statement parameters for the specified files:

- **MODE=FAIL**
  SECURITRE will run files 1, 5-25 in FAIL mode. A security violation will occur if the user is not authorized access to these files.

- **NAME=PERSONEL**
  SECURITRE will use the file name alias of PERSONEL for files 1 and 5-25 in place of their file numbers when generating DSNs.

- **NOIDRED=ACCEPT**
  If SECURITRE is processing a READ command, such as L3 or S1, against files 1, 5-25 and it cannot obtain a User-ID, SECURITRE will still allow the command to be processed.

- **NOIDUPD=REJECT**
  If SECURITRE is processing an UPDATE command, such as A1 or E1, against files 1, 5-25 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed.

(continued on next page)

(continued from previous page)

The STRFNR statement for **FILE=100** indicates the following parameters will override the previously defined STRDEF statement parameters for this file:

- **LOGVIOL=ALL**
  SECURITRE will cause the SSF to log all violations in the system security log for each user denied access to file 100.

- **MODE=FAIL**
  SECURITRE will run file 100 in FAIL mode. A security violation will occur if the user is not authorized access to this file.

- **NAME=PAYROLL**
  SECURITRE will use the file name alias of PAYROLL for file 100 in place of its file number when generating DSNs.

- **NOIDRED=REJECT**
  If SECURITRE is processing a READ command, such as L3 or S1, against file 100 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed.

- **NOIDUPD=REJECT**
  If SECURITRE is processing an UPDATE command, such as A1 or E1, against file 100 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed.

The STRFNR statement for **FILE=101** indicates the following parameters will override the previously defined STRDEF statement parameters for this file:

- **DSNORDR=(JOB, NLIB, NPGM, GPGM, FILE)**
  For File Security on file 101, the DSN generated will include the MVS Jobname, the NATURAL library, and the NATURAL program name if the call comes from NATURAL or the non-NATURAL program name if the call did not come from NATURAL and the literal "SALARY" to replace the file number (101). For example, if the call comes from the NATURAL program PAYRAISE, in the PAYLIB library, SECURITRE will generate and check authorization for the following DSN:

  ```
  ADABAS.PROD.CICSPROD.PAYLIB.PAYRAISE.SALARY
  ```

  If the call comes from a COBOL program PAYBOOST that was run in batch job JOEBATCH, SECURITRE will generate and check the following DSN:

  ```
  ADABAS.PROD.JOEBATCH.PAYBOOST.SALARY
  ```

- **LOGVIOL=ALL**
  SECURITRE will cause the SSF to log all violations to the system security log for each user denied access to file 101.

- **MODE=FAIL**
  SECURITRE will run file 101 in FAIL mode. A security violation will occur if the user is not authorized access to this file.

- **NAME=SALARY**
  SECURITRE will use the file name alias of SALARY for file 101 in place of its file number when generating DSNs.

- **NOIDRED=REJECT**
  If SECURITRE is processing a READ command, such as L3 or S1, against files 101 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed.

- **NOIDUPD=REJECT**
  If SECURITRE is processing an UPDATE command, such as A1 or E1, against file 101 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed.

The STRFNR statement for **FILE=(150-155)** indicates that the following parameters will override the previously defined STRDEF statement parameters for the specified files:

- **DSNORDR=JOB**
  If a user accesses files 150-155, the DSN will only include the PREFIX, QUALIFY, and MVS Jobname. SECURITRE will generate the following DSN:

  ```
  ADABAS.PROD.CICSPROD
  ```

- **MODE=FAIL**
  SECURITRE will run files 150-155 in FAIL mode. A security violation will occur if the user is not authorized access to these files.

- **NAME=PRODFILE**
  SECURITRE will use the file name alias of PRODFILE for files 150-155 in place of their file numbers when generating DSNs.

The STRFNR statement for **FILE=900** indicates the following parameters will override the previously defined STRDEF statement parameters for this file:

- **MODE=WARN**
  SECURITRE will operate file 900 in WARN mode. This is a new file that is likely to be secured in the future. File access will be allowed, but a violation will be logged if the user is not authorized to access to these files.

- **NAME=NEWFILE**
  SECURITRE will use the file name alias of NEWFILE for file 900 in place of its file number when generating DSNs.

Example 2:
Assume the following parameters:

```
                                                                      Col. 72
STRDEF    CLASS=DATASET,CMDLOG=ON,DELIM='.',DSNORDR=(FILE),
          DSNPOOL=100,FLSDEL=DEL,FLSPOOL=10,FORCE=99,                   x
          LOGVIOL=ALL,MODE=DORMANT,                                     x
          NOIDRED=REJECT,NOIDUPD=REJECT,PREFIX='ADABAS',                x
          PURINTT=3600,PURINTV=8,QUALIFY='',                            x
          RTMORDR=DBID,PROCCL=OFF,SECURE=ACF2,                          x
          STREX2=SBVEXIT,STRRTM='ADABAS.STR',                           x
          USERID=STRUEXB,USERS=50,UTMODE=WARN,                          x
          UTORDER=(UTIL),UTPREF='ADABAS.UTL'
*
STRFNR    FILE=100,NAME='CNFDNCL',MODE=FAIL,                            x
          DSNORDR=(FILE,TERM)
*
STRFNR    FILE=150,NAME='PAYROLL',MODE=FAIL,FLSMODE=WARN,               x
          FIELDS=(CK,SALARY,CM,RATECAT),DSNORDR=(FILE,FIELD)
*
STRFNR    FILE=200,NAME='SECRET',MODE=FAIL
```

The following STRDEF parameters were not included, so SECURITRE will assign the default values indicated:

```
FILEMAX=OLD
PRINT=NOGEN
TRACE=OFF
USERID2=NONE
STREX1=                    (none)
TRMRTM=CONTROL.TRM
TERM=S                     (stop)
UEXIT1=                    (none)
```

The STRFNR END parameter is not needed, because FILEMAX=OLD default is used, and no STRFNR parameters are defined for a file **> 255**.

- **CLASS='DATASET'**
  SECURITRE will use the class "DATASET" for creating the first part of the DSN when making calls to the SSF.

- **CMDLOG=ON**
  If no other User-Exit-4s are called by SECURITRE, SECURITRE will flag all commands to be logged by ADABAS.

- **DELIM='.'**
  SECURITRE will use a period as the delimiter character between all DSN components when DSNs are generated for File Security, Utility Security, and RTM Security.

(continued on next page)

(continued from previous page)

- **DSNORDR=FILE**
  If a user accesses any file except files 100 and 150, the DSN will only include the PREFIX and file number or STRFNR file alias NAME override. For example, SECURITRE will generate and check authorization to the following DSN for file 160:

  ```
  ADABAS.F160
  ```

- **DSNPOOL=100**
  SECURITRE will allocate space for 100 DSNs in its DSNPOOL table. If the site had taken only File Security into account and only 4 terminals were used to access file 100, they would have expected only about 5 "legal" DSNs to be generated. However, this site also uses SECURITRE for NATURAL, and has included those DSNs in estimating the DSNPOOL size.

- **FLSDEL='DEL'**
  SECURITRE will use the literal 'DEL' in place of the field alias on field level security DSNs when an ADABAS E1 or E4 command is being processed.

- **FLSPOOL=10**
  SECURITRE will maintain Field Level Security information for up to 10 CIDs per user in its tables.

- **FORCE=99**
  SECURITRE will not perform a daily table purge.

- **LOGVIOL=ALL**
  SECURITRE will cause the SSF to log all violations for File Security.

- **MODE=DORMANT**
  All files except 100 and 200 remain unsecured, so DSNs will not be generated for these files.

- **NOIDRED=REJECT, NOIDUPD=REJECT**
  If SECURITRE is processing a READ or UPDATE command against files 100 or 200 and it cannot obtain a User-ID, SECURITRE will not allow the command to be processed. For all other files, it will process the command because MODE=DORMANT.

- **PROCCL=OFF**
  SECURITRE will not delete user information from its tables when a user issues a CL command.

- **PURINTT=3600**
  Users who have not issued a command for more than 1 hour (3600 seconds) will be considered "inactive" by SECURITRE when a purge takes place.

- **PURINTV=8**
  SECURITRE will purge its tables of inactive users every 8 hours.

- **QUALIFY=''**
  SECURITRE will not use a qualifier when making calls to the SSF.

(continued on next page)

(continued from previous page)

- **RTMORDR=DBID,STRRTM='ADABAS.STR'**
  When a user executes a secured RTM function, SECURITRE will generate the DSN as follows:

  ```
  ADABAS.STR.Dnnn or ADABAS.STR.Dnnnnn
  ```

  In this case, 'nnn' and 'nnnnn' are the database numbers. For example, for all RTM accesses of database 42, SECURITRE will generate and check the following DSN:

  ```
  'ADABAS.STR.D042'
  ```

  Note: Using only the DBID in the RTMORDR will permit authorized users to that DSN to run all RTM functions on that database.

- **STREX2=SBVEXIT**
  After a user's first call has passed File and Field Level Security, SECURITRE will call the program SBVEXIT to obtain the user's ADABAS password.

- **SECURE=ACF2**
  The SSF in use at this site is ACF2.

- **USERS=50**
  This site expects about 45 users to use the database during peak periods. They are requesting that SECURITRE allocate space for 50 users. By padding this parameter, they will avoid the possibility of running out of space.

- **USERID=STRUEXB**
  SECURITRE will look for a USERINFO Area that it has generated in order to find the User-ID and other information. The site has not set up any alternate method to obtain a User-ID.

- **UTMODE=WARN**
  SECURITRE will run Utility Security in WARN mode. The user will be allowed execution of the utility, but a violation will be logged if the user is not authorized to access the utility. This site is still in the process of setting up Utility Security and does not want jobs that are not yet authorized to ABEND.

- **UTORDER=(UTIL),UTPREF='ADABAS.UTL'**
  SECURITRE Utility Security will generate the DSNs as follows:

  ```
  ADABAS.UTL.uuu
  ```

  In this case, 'uuu' refers to the last three characters of the utility. For example, when a user runs the ADAULD utility function, Utility Security will generate and check the following DSN:

  ```
  ADABAS.UTL.ULD
  ```

The STRFNR statement for **FILE=100** indicates that the following parameters will override the previously defined STRDEF statement parameters for this file:

- **DSNORDR=(FILE,TERM)**
  For file 100, if the call is coming from CICS, the DSN will only include the PREFIX file number or STRFNR file alias NAME override and the Terminal-ID. For example, SECURITRE will generate the DSN as follows:

  ```
  ADABAS.CNFDNCL.term
  ```

  If the call is coming from TSO or BATCH, 'term' refers to the CICS Terminal-ID, SECURITRE will generate and check authorization to the following DSN:

  ```
  ADABAS.CNFDNCL
  ```

- **MODE=FAIL**
  SECURITRE will run file 100 in FAIL mode. A security violation will occur if the user is not authorized access to this file.

- **NAME=CNFDNCL**
  SECURITRE will use the file name alias of CNFDNCL for file 100 in place of its file number when generating DSNs.

The STRFNR statement for **FILE=150** indicates the following parameters are in effect for calls to this file.

- **DSNORDR=(FILE,FIELD)**
  When a call is made against File 150, SECURITRE will first generate a DSN with the PREFIX, QUALIFY, and file name (ADABAS.PAYROLL). If the user is allowed access to that DSN, SECURITRE will then ask if the user has access for a second DSN with the PREFIX, QUALIFY, file name, and field alias for each secured field that is requested in the Format Buffer:

  ```
  'ADABAS.PAYROLL.SALARY and/or DABAS.PAYROLL.RATECAT
  ```

- **FIELDS=(CK,SALARY,CM,RATECAT)**
  If the field CK is listed in the Format Buffer, SECURITRE will use the alias "SALARY" in the DSN as shown above. If the field CM is listed in the Format Buffer, SECURITRE will use the alias "RATECAT" in the DSN.

- **FLSMODE=WARN**
  Field Level Security is in WARN mode for this file. If the user asks for data from any of the fields listed in the FIELDS parameter, SECURITRE will ask the SSF if the user has access to the field(s). Any violations will be logged, and the user will be allowed access.

- **MODE=FAIL**
  SECURITRE will run file 150 in FAIL mode. If the user is denied access to this file, a security violation will be logged, and Field Level Security processing will not take place.

- **NAME='PAYROLL'**
  The alias PAYROLL will be included in the DSN in place of the file number when a DSN is built.

The STRFNR statement for **FILE=200** indicates the following parameters will override the previously defined STRDEF statement parameters for this file:

- **MODE=FAIL**
  SECURITRE will run file 200 in FAIL mode. A security violation will occur if the user is not authorized access to these files.

- **NAME=SECRET**
  SECURITRE will use the file name alias of SECRET for file 200 in place of its file number when generating DSNs.

Example 3:
Assume the following parameters:

```
                                                                      Col. 72
STRDEF  DELIM='',DSNORDR=FILE,FILEMAX=NEW,FLSPOOL=0,PREFIX='ADA',       x
QUALIFY='',RTMORDR=DBID,SECURE=RACF,STRRTM='ADA',                       x
USERID=TRIMV4-1,UTMODE=DORMANT                                          x

*
STRFNR  FILE=(1-200),MODE=DORMANT
*
STRFNR  FILE=(201-982),NAME='PAYR',MODE=FAIL
*
STRFNR  END
```

The following STRDEF parameters were not included, so SECURITRE will assign the default values indicated:

```
CLASS='DATASET'          CMDLOG=OFF                FLSDEL=DELETE
FORCE=99  (never)        LOGVIOL=ALL               MODE=FAIL
NOIDRED=REJECT           NOIDUPD=REJECT            PRINT=NOGEN
PROCCL=ON                PURINTT=0                 PURINTV=0  (do not purge)
RTMORDR=(FUNC,DBID)      STREX1=   (none)          STREX2=    (none)
TERM=S    (stop)         TRACE=OFF                 TRMRTM=CONTROL.TRM
UEXIT1=   (none)         USERID2=NONE              USRPOOL=400
USERS=100                UTORDER=(UTIL,FUNC,FILE)  UTPREF='ADAUTIL'
```

- **PREFIX='ADA',DELIM='',QUALIFY=''**
  SECURITRE will generate the beginning of the DSN as 'ADA'.

- **DSNORDR=FILE**
  If a user accesses any file, the DSN will only include the PREFIX and file number or the STRFNR file alias NAME override. For example, SECURITRE will generate the following DSN for file 100:

  ```
  ADAF100
  ```

- **FILEMAX=NEW**
  SECURITRE will generate a variable number of 80-byte file entries dependent on the number of files with STRFNR parameters.

- **FLSPOOL=0**
  SECURITRE will not allocate space for Field Level Security processing.

- **RTMORDR=DBID,STRRTM='ADA'**
  SECURITRE will generate the DSNs for RTM Security as follows:

  ```
  ADADnnn
  ```

  In this case, 'nnn' represents the database number. For example, for all RTM accesses of database 42, RTM Security will generate and check the following DSN:

  ```
  ADAD042
  ```

- **SECURE=RACF**
  The SSF in use at this site is RACF.

- **USERID=TRIMV4-1**
  SECURITRE will obtain the User-ID from Additions-3 of the Control Block, according to USERID. The appropriate changes to ADALNC, ADALNK, ADALCO, and/or LNKOLM/LNKOLSC have been made at installation time. SECURITRE cannot use any information in the DSN besides the file name, since it is obtaining the User-ID from Additions-3 instead of the USERINFO Area.

- **UTMODE=DORMANT**
  Utility Security will not be invoked even if the SECURITRE Utility Security front-end has been linked with ADARUN.

The STRFNR statement for **FILE=(1-200)** indicates the following parameters will override the previously defined STRDEF statement parameters for the specified files:

- **MODE=DORMANT**
  No file security checking will take place for files 1-200, which are in DORMANT mode.

The STRFNR statement for **FILE=(201-982)** indicates that the following parameters will override the previously defined STRDEF statement parameters for the specified files:

- **MODE=FAIL**
  SECURITRE will run files 201-982 in FAIL mode. A security violation will occur if the user is not authorized access to these files.

- **NAME=PAYR**
  SECURITRE will use the file name alias of PAYR for files 201-982 in place of their file numbers when generating DSNs. For example, SECURITRE will combine the PREFIX and STRFNR file alias NAME override with no qualifier and no delimiter and generate the following DSN:

```
ADAPAYR
```

SECTION III

SETTING UP SECURITRE FOR NATURAL

III.1 Introduction

Within the NATURAL application environment, there is a need to provide security to control:

- The ability to access NATURAL for a given set of FDIC, FNAT, and FUSER files
- The ability to logon to certain NATURAL application libraries
- The execution of particular programs in the application
- The ability to read program source, catalog, and save programs in the application
- The ability to RUN NATURAL programs from within an application
- The use of DDMs

Just as SECURITRE User-Exit-1 to ADABAS permits all security checks on ADABAS database, file, and field levels to be made by the SSF, SECURITRE for NATURAL allows security checks to be put into place to control the critical NATURAL resources.

III.2 SECURITRE for NATURAL

Six levels of security are provided by SECURITRE for NATURAL: NATURAL Session Initialization, LOGON, DDM, PROGRAM, RUN Security, and NATURAL Utility Security.

Each of these levels of security may run in DORMANT, WARN, or FAIL mode. For example, NATURAL Session Initialization may run in FAIL mode while DDM Security is in WARN mode and LOGON Security is in DORMANT mode.

The security checks, which take place at NATURAL Session Initialization (NSI) time, determine whether the user may enter the NATURAL environment using the FNAT, FUSER, and FDIC files specified in their NATURAL startup parameter specifications (NATPARM module). Using the STNPARM parameters, SECURITRE generates three DSNs, one each for the FNAT, FUSER, and FDIC files that it sends to the SSF for authorization. If NSI Security is in FAIL mode and the SSF denies the request, the user will be denied access to NATURAL.

LOGON Security is used to determine which users may logon to a library. When a user executes the LOGON command, SECURITRE will generate a DSN based on the STNPARM and STNLIB parameters and request authorization from the SSF. If the request is accepted, SECURITRE will allow the logon and will execute the startup program that was specified in the STNLIB parameters for that library. If the request is denied, SECURITRE will prevent the logon from taking place and will allow the user to specify another library.

If DDM Security is in WARN or FAIL mode, a DSN will be generated when a user attempts to "read a DDM" (i.e., compile a program that references the DDM). A DSN will be generated using the STNPARM and STNDDM parameters, and a request for authorization to that DSN will be sent to the SSF. If access is denied and DDM Security is in FAIL mode, the user will be prevented from viewing or using the DDM (i.e., the program will not compile).

PROGRAM Security may be turned on or off at the library level. There are four types of Program Security: program source read (READ or EDIT), source write (SAVE), object read (EXECUTE), and object write (CAT). If PROGRAM Security is in WARN or FAIL mode for a given library, SECURITRE will detect these types of program accesses and will generate a DSN based on the STNPARM parameters. If the request is denied and PROGRAM Security is in FAIL mode, the user will be prevented from doing such things as editing, saving, storing, and executing the program. Since the PROGRAM Security mode is specified at the library level, a site may secure programs in some libraries but not others.

In addition, Program Execution Security may be turned on for specific types of programs or objects. For example, a site may choose to have SECURITRE check security for programs and subroutines but not for subprograms, maps, and help routines.

Program write security may be used to determine whether a specific user may SAVE/CAT/STOW/PURGE any NATURAL objects while logged on to a library. This check is made once, when the user logs on to a library, and does not need to be made for any individual object.

RUN Security may also be specified at the library level. When a user issues a RUN command, SECURITRE will generate a DSN based on the STNPARM parameters and request authorization for the DSN. If access is denied, the RUN command will not execute.

NATURAL Utility Security may be used to prevent unauthorized users from executing specific NATURAL utilities. When a user attempts to execute a NATURAL utility, such as SYSMAIN or SYSERR, SECURITRE will generate a DSN based on the STNPARM parameters and request authorization for the DSN. If access is denied, SECURITRE will prevent the utility from executing.

For each of these types of security, SECURITRE provides an ORDER parameter, enabling a site to specify which pieces of information should be included in a DSN and the order they are to appear.

III.3 Setting Up SECURITRE for NATURAL Parameters

SECURITRE for NATURAL has many parameters that make it very flexible. Understanding all aspects of the parameters may take some time. When SECURITRE for NATURAL is initially installed, special attention should be given to the parameters discussed below. These parameters include SERVER, DELIM, PREFIX, QUALIFY, PRIVBUF, and PGMTBSZ, which are defined in the STNPARM statement. Once SECURITRE for NATURAL is running, fine-tuning adjustments may be made to other parameters to tailor SECURITRE for NATURAL to the site's specific needs.

SERVER

The SERVER parameter specifies a database that is running the SECURITRE User-Exit-1. By directing SSF calls through a server, the NATURAL module does not need to run from an APF-Authorized dataset. The DSNs that are passed from SECURITRE for NATURAL are stored in the User-Exit-1 tables for more efficient processing.

A non-zero DBID should be assigned to the SERVER parameter. If the SERVER database is not running SECURITRE or if it is not currently active, all calls will be treated as if they were rejected by the SSF. Therefore, the SERVER should point to a database that is consistently up and running with SECURITRE User-Exit-1. Assigning a non-production database to the SERVER parameter can reduce overhead on the production database.

DELIM and QUALIFY

Both the DELIM and QUALIFY parameters may be set to null, specified as '' (two single quotes). This is useful if very short DSNs are required or if the QUALIFY parameter is regarded as unnecessary for processing at a particular site.

USER BUFFER

SECURITRE allows flexibility as to where to allocate space for each user's buffer when a user enters NATURAL. The default is to use the USER BUFFER, but a simple zap can cause the space to be allocated in any other NATURAL buffer or STNGET/STNFREE can do a GETMAIN/FREEMAIN combination.

SECURITRE calls the module STNFREE when a FIN is executed from NATURAL. STNFREE is used to de-allocate the user's buffer.

MODE Parameters

The different MODE parameters specify the type of security required for each component of SECURITRE for NATURAL. When first setting up SECURITRE for NATURAL, the site may want to set the following mode parameters to DORMANT or WARN:

```
DDMMODE      (defined by the STNPARM statement)
             (interacts with STNDDM TYPE)
LGNMODE      (defined by the STNPARM statement)
             (interacts with STNLIB TYPE)
NSIMODE      (defined by the STNPARM statement)
NUMODE       (defined by the STNPARM statement)
PGMCHK       (defined by the STNLIB statement)
PGWRTCK      (defined by the STNLIB statement)
RUNCHK       (defined by the STNLIB statement)
```

If these parameters are set to DORMANT or WARN, access to NATURAL will only be prohibited if there is a problem with the PRIVBUF/NATPARM parameter combination.

Initially, it may be useful to set the SERVER parameter to a DBA-only type of test database and set the above MODE parameters to WARN. If the SECURITRE STRDEF TRACE parameter is turned ON in the SERVER database, SECURITRE will print the DSNs being sent to the SERVER for verification. The TRACE will help sites verify that DSNs are being created according to expectations. TRACE should only be turned ON for a limited time, and only if there is a small number of users accessing the database in question. Otherwise, SECURITRE will generate an excessive amount of output.

Sample SECURITRE for NATURAL parameters can be found in the SOURCE dataset on the SECURITRE release tape. For more information about these parameters, refer to the SECURITRE Reference Manual.

III.4 Controlling Access to NATURAL

SECURITRE for NATURAL can be used at NATURAL Session Initialization (NSI) time to determine whether or not a user may access NATURAL using the FDIC, FNAT, and FUSER system files indicated in the NATURAL startup parameter specifications (NATPARM module).

If the STNPARM parameter NSIMODE is set to DORMANT, calls to the SSF will not be made for NSI Security.

If the STNPARM parameter NSIMODE is set to WARN or FAIL, three DSNs will be generated and three SSF calls will be made, one for each of the three system files mentioned above. These DSNs will be generated according to the order specified in the NSIORDR parameter.

For example, assume that the following parameters have been included in the parameter module:

```
                                                                    Col. 72
STNPARM         PREFIX='NAT',QUAL='PROD',DELIM='.',                    x
                NSIMODE=FAIL,NSIORDR=(LIT,FILE),                       x
                NSIFDIC=NSIFDIC,NSIFNAT=NSIFNAT,NSIFUSR=NSIFUSR
STNFILE         PRODFDIC,DBID=123,FNR=003
STNFILE         PRODFNAT,DBID=123,FNR=002
STNFILE         PRODFUSR,DBID=123,FNR=001
```

Then the three DSNs generated will be:

```
NAT.PROD.NSIFDIC.PRODFDIC
NAT.PROD.NSIFNAT.PRODFNAT
NAT.PROD.NSIFUSR.PRODFUSR
```

When attempting to access NATURAL, SECURITRE will make a request to the SSF to check whether or not the user has access to these dataset names. If the request is denied and NSIMODE=FAIL, the user will not be allowed to invoke NATURAL. If the request is denied and NSIMODE=WARN, the user will be allowed to invoke NATURAL, but a message will appear in the system security log indicating that a security violation occurred. If the request is accepted, the user will be allowed to invoke NATURAL and further checks on Logon, Programs, and DDMs may be made.

III.5 Controlling Logon Access to NATURAL Application Libraries

SECURITRE for NATURAL may be used to determine if the users can logon to a particular application. LOGON Security may be tailored by using the STNPARM parameters LGNMODE, LGNLIT, LGNORDR, and LGNPRIV, as well as the STNLIB parameter TYPE. Each of these is described in the Reference Manual.

LOGON security checking is implemented using the NATURAL program STRLOGON, which calls an Assembler program to check security and assign parameters. The program then fetches the Software AG LOGON.

In order to implement STRLOGON, the real "LOGON" program in SYSLIB is renamed STRLOGON, and the STRLOGON program, which is part of SECURITRE, is copied to SYSLIB and named LOGON.

If desired, sites may customize their own LOGON program to use in place of the STRLOGON supplied with SECURITRE. The security is handled through a call to an assembler module STRLGN. STRLGN is called using the following parameters:

```
LIB                 (A8)       /* the library to logon to
PRIVILEGE           (A1)       /* "R": Read  or  "U": Update
RETCD               (A1)       /* "Y": ok    or  "N": not allowed
STARTUP             (A8)       /* the startup trans for the library
LGNPARMS            (A16)      /* miscellaneous information
```

Normally, PRIVILEGE will be set to "R" before the call, requesting normal logon privileges. If "U" is specified, and the call is accepted, NO SECURITY CHECKING WILL TAKE PLACE WHILE THE USER IS LOGGED ON TO THE LIBRARY. That is, the user may access all DDMS and all programs from other libraries using commands, such as EDIT PAYRAISE PAYLIB. Therefore, "U" should be used for DBAs and other individuals who might require special privileges.

For all others, it is recommended that all DSNs used by SECURITRE for NATURAL should be set up in the SSF for READ access ONLY. When the return code specifies that a logon is allowed, the new logon program can STACK TOP COMMAND #STARTUP and then FETCH 'STRLOGON' in order to execute the real LOGON. If the logon request is denied, the new LOGON program can handle the situation accordingly. A sample STRLOGON program and STRLOGN map are included in the NATLOAD dataset on the release tape.

The site may use the STNLIB parameter LGNPRMS to customize the NATURAL environment at LOGON time. LGNPRMS is a 16-byte area that is passed to the STRLOGON program when STRLGN is called. For example, if lower case were to be used in all libraries except ALIB, then the STNLIB statements would look like

```
STNLIB *DEFAULT,LGNPRMS='L'
STNLIB ALIB,LGNPRMS='U'
```

and STRLOGON would include the code

```
REDEFINE #LGNPRMS (#LGN·CASE (A1))
    .
    .
    .
CALL 'STRLGN' #PARMLIST
IF #RETCD='Y'
    THEN SET CONTROL #LGN·CASE
```

When LOGON Security is used, SECURITRE generates a DSN that contains all of the components specified in the LGNORDR parameter. These consist of the library name, the logon literal, and the FUSER being used. If the STNFILE alias for the FUSER is included in LGNORDR the alias for FUSER will be generated as part of all DSNs. This includes system libraries (SYS*) that are stored in the FNAT file. For example, assume the following parameters:

```
                                                                  Col. 72
STNPARM          PREFIX='NAT',QUALIFY='PROD',LGNMODE=FAIL,          x
                 LGNLIT=LGN,LGNORDR=(LIB,LIT,FUSER)
STNLIB           PAYROLL,TYPE=PRIV
STNFILE          PRODFUSR,DBID=100,FNR=200
```

If the user is trying to logon to the production PAYROLL library, the DSN generated will be:

```
NAT.PROD.PAYROLL.LGN.PRODFUSR
```

If LGNORDR=(LIT,LIB) is specified, the following DSN will be generated:

```
NAT.PROD.LGN.PAYROLL
```

Some sites want to secure particular libraries, but allow users to logon to their own "Private" libraries. SECURITRE may be set up to bypass LOGON Security for private libraries, depending on the value specified in the STNPARM parameter LGNPRIV. By using the LGNPRIV parameter, a site can keep LOGON Security in FAIL mode with the TYPE for *DEFAULT=PRIV, but it does not have to code STNLIB statements for each individual User-ID.

Example 1:
Assume the following parameters:

```
                                                                  Col. 72
STNPARM          PREFIX='NATURAL',                                  x
                 QUALIFY='PROD',                                    x
                 DELIM='.',                                         x
                 LGNMODE=FAIL,                                      x
                 LGNLIT=LGN,                                        x
                 LGNORDR=(LIT,LIB)
STNLIB           *DEFAULT,TYPE=PRIV
STNLIB           PUBLIC,TYPE=PUB
STNFILE          PRODFUSR,DBID=123,FNR=001
```

When a user attempts to logon to the PAYLIB library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PROD.LGN.PAYLIB
```

If LGNORDR=(FUSER,LIB,LIT) is specified, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PROD.PRODFUSR.PAYLIB.LGN
```

If the user attempts to logon to the PUBLIC library, the DSN will not be generated because the STNLIB parameter TYPE=PUB.

Example 2:
Assume the following parameters:

```
                                                                  Col. 72
STNPARM        PREFIX='NATURAL',                                    x
               QUALIFY='PROD',                                      x
               DELIM='.',                                           x
               LGNMODE=FAIL,                                        x
               LGNLIT=LGN,                                          x
               LGNORDR=(LIT,LIB),                                   x
               LGNPRIV=UID+
STNLIB         *DEFAULT,TYPE=PRIV
```

When user FRED attempts to logon to FRED library, no security checking is performed. Likewise, user FRED may logon to FREDA, FRED123, etc. without being checked for security.

However, when user FRED attempts to logon to the PAYROLL library, SECURITRE will check with the SSF to see if FRED is authorized to access the following DSN because LGNMODE=FAIL and the default TYPE=PRIV:

```
NATURAL.PROD.LGN.PAYROLL
```

If LGNPRIV=UID is specified, FRED would be allowed to logon to the library FRED with no security checking; but when FRED attempts to logon to the FREDLIB library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PROD.LGN.FREDLIB
```

LIBFUSR and other Library Level Parameters
If a site uses the same NATURAL module for users on different FUSERs (e.g., if Test users and Production users both use the same NATURAL) the LIBFUSR parameter can be used to designate that certain STNLIB statements should be used for particular library/FUSER combinations. The value assigned to the LIBFUSR parameter should be the STNFILE alias for a DBID/FNR combination.

For example, if payroll programs are to be secured at the production level but not the test level, a group of STNLIB parameters can be set as follows:

```
STNLIB PAYLIB,LIBFUSR=PRODFUSR,PGMCHK=FAIL
STNLIB PAYLIB,LIBFUSR=TESTFUSR,PGMCHK=DORMANT
```

Leaving a blank LIBFUSR parameter in an STNLIB statement specifies that the STNLIB should be used as a default for this library.

Example 3:
Assume the following parameters:

```
STNPARM    LGNMODE=FAIL                                                    (1)
STNLIB     *DEFAULT,LIBFUSR=PROD,TYPE=PRIV,PGMCHK=DORMANT                  (2)
STNLIB     *DEFAULT,LIBFUSR=STG,TYPE=PRIV,PGMCHK=DORMANT                   (3)
STNLIB     *DEFAULT,LIBFUSR=TEST,TYPE=PRIV,PGMCHK=DORMANT                  (4)
STNLIB     *DEFAULT,TYPE=PUB,PGMCHK=DORMANT                                (5)
STNLIB     PAYROLL,LIBFUSR=PROD,TYPE=PRIV,PGMCHK=FAIL                      (6)
STNLIB     PAYROLL,LIBFUSR=STG,TYPE=PRIV,PGMCHK=WARN                       (7)
STNLIB     PAYROLL,LIBFUSR=TEST,TYPE=PRIV,PGMCHK=WARN                      (8)
STNLIB     PAYROLL,TYPE=PUB,PGMCHK=DORMANT                                 (9)
STNFILE    PROD,DBID=100,FNR=101                                          (10)
STNFILE    STG,DBID=150,FNR=101                                           (11)
STNFILE    TEST,DBID=175,FNR=101                                          (12)
STNFILE    DBA,DBID=200,FNR=101                                           (13)
```

At this site there are four databases: PROD, STG, TEST, and DBA. When a user is accessing the FUSER on the DBA database, the STNLIB statement on either line (5) or (9) will be referenced, depending on which library they are accessing. For the DBA FUSER, SECURITRE selects the statements with the blank LIBFUSR because no statements have been coded with LIBFUSR=DBA.

If a user is accessing the FUSER on any of the other three databases, SECURITRE will refer to the STNLIB statements with LIBFUSR matching the user's FUSER. For example, if a user specified FUSER=(150,101), the STG FUSER, in the NATPARMs, SECURITRE would use the STNLIB on line (7) for the PAYROLL library and the STNLIB on line (3) for all other libraries.

Program Security will be in FAIL mode for the PROD PAYROLL library. It will be in WARN mode for the STG and TEST PAYROLL libraries. It will be DORMANT mode for all other libraries, including lines (2)-(9).

If a user logs on to the PUBLIC library in production, line (2) will be selected for the STNLIB parameters pertaining to the LOGON. If the user logs on to the PAYROLL library in production, line (6) will be selected. If they logon to the PUBLIC library on the DBA database, line (5) will be selected.

III.6 Controlling Access to NATURAL Programs

There are four types of Program Security that may be controlled by SECURITRE for NATURAL. These are: program source read (READ or EDIT), program source write (SAVE), program object read (EXECUTE), and program object write (CAT).

Whether or not program checking takes place is determined at the library level through the STNLIB parameter PGMCHK. If PGMCHK is set to WARN or FAIL and one of the above types of access mentioned above is attempted by the user, SECURITRE will generate a DSN according to the guidelines provided in the STNPARM parameters PGLITOR (object read literal), PGLITSR (source read literal), PGLITOW (object write literal), PGLITSW (source write literal), and PGMORDR (order to include the DSN items).

The components that may be included in a DSN are the literal, the library, the program, and the FUSER file being used. Once the DSN is generated, SECURITRE will send a request to the SSF to determine whether the user is allowed access to the DSN or not. If access is not allowed, the user will not be able to do such things as EDIT or SAVE to the program. For more information about the types of errors generated, refer to **Section X - Operations Considerations**.

For example, assume the following parameters:

```
                                                                  Col. 72
STNPARM         PGLITOR=EXEC,PGLITSR=READ,                          x
                PGLITOW=STOW,PGLITSW=SAVE,                          x
                PGMORDR=(FUSER,LIT,LIB,PGM),PREFIX='NAT',           x
                QUALIFY='PROD'
STNLIB          PAYROLL,PGMCHK=FAIL
STNFILE         PRODFUSR,DBID=123,FNR=1000
```

For program PAYRAISE in the PAYROLL library, depending on the type of access taking place, one of the following DSNs will be generated:

```
NAT.PROD.PRODFUSR.EXEC.PAYROLL.PAYRAISE
NAT.PROD.PRODFUSR.READ.PAYROLL.PAYRAISE
NAT.PROD.PRODFUSR.STOW.PAYROLL.PAYRAISE
NAT.PROD.PRODFUSR.SAVE.PAYROLL.PAYRAISE
```

Including the FUSER as part of the DSN allows the site to specify, for example, that the program PAYRAISE may be executed by a particular user from the production system, but not from the test system.

In order to reduce the number of calls SECURITRE makes to the SSF, SECURITRE maintains a table of programs that have been accepted for execution. This table allows SECURITRE to "remember" which programs may be accessed. Subsequent calls to access the same programs will incur little overhead by eliminating the need to call the SSF. The number of programs to be maintained in this table is specified in the PGMTBSZ parameter. Should the number of accepted programs exceed the number specified in PGMTBSZ, the oldest entry will be deleted to make room for the most recently accessed program. When a LOGON takes place, the table will be cleared.

If particular users are to be completely restricted from writing programs from a given library, the PGWRTCK parameter may be used to determine each user's ability to write programs. After LOGON security checking takes place, SECURITRE will generate a DSN using the PGWLIT and PGWORDR parameters if PGWRTCK is WARN or FAIL. This DSN is sent to the SSF to determine whether the user is allowed access to the DSN. If access is permitted, the user will be allowed to SAVE/CAT/STOW/SCRATCH/PURGE programs while logged on to that library, subject to the restrictions placed by program security. If access is not permitted, the user will be prevented from writing any source or object while logged on to the library, and program security will not be checked for source write or object write.

For example, assume the following parameters:

```
                                                                      Col. 72
STNPARM        PGWLIT=PWRITE,PGMORDR=(LIT,LIB),                          x
               PREFIX='NATURAL',QUALIFY='PROD',DELIM='.'
STNLIB         PAYROLL,PGWRTCK=FAIL
```

When a user logs on to the PAYROLL library, SECURITRE will generate and check authorization to the DSN below:

```
NATURAL.PROD.PWRITE.PAYROLL
```

III.7 Controlling Access to DDMs

The purpose of DDM Security is to provide protection against unauthorized users obtaining information about what fields are defined in a file and from users using a DDM for which they are not authorized in a program that they are writing. Using DDM Security alone does not protect the data.

SECURITRE is able to detect attempts to access DDMs. Access to a DDM occurs when a user tries to view a DDM in NATURAL, run a program, or compile a program. If DDM Security is requested through the STNPARM parameter DDMMODE, a DSN will be generated according to the values in the parameters DDMLIT and DDMORDR, and SECURITRE will send a request to the SSF for authorization. Other DDM-related parameters include ALIAS and TYPE which are defined in the STNDDM statement. For more information about these parameters, refer to the SECURITRE Reference Manual.

Example 1:
Assume the following parameters:

```
                                                                  Col. 72
STNPARM        PREFIX='NATURAL',                                    x
               QUALIFY='PROD',                                      x
               DELIM='.',                                           x
               DDMLIT=DDM,                                          x
               DDMMODE=FAIL,                                        x
               DDMORDR=(LIT,LIB,DDM)
STNDDM         PAYROLL-SALARY,ALIAS=SALARY,TYPE=PRIV
```

When a user attempts to access the DDM PAYROLL-SALARY while logged on to the PAYLIB library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PROD.DDM.PAYLIB.SALARY
```

If DSNORDR=(DDM) is specified, SECURITRE will generate and check the following DSN:

```
NATURAL.PROD.SALARY
```

If DSNORDR=(LIB,DDM) is specified, SECURITRE will generate and check the following DSN:

```
NATURAL.PROD.PAYLIB.SALARY
```

Example 2:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM        PREFIX='NATURAL',                                      x
               QUALIFY='',                                            x
               DELIM='.',                                             x
               DDMLIT=DDM,                                            x
               DDMMODE=FAIL,                                          x
               DDMORDR=(FDIC,LIT,LIB,DDM)
STNDDM         PAYROLL-SALARY,DDMNAME=YES,TYPE=PRIV
STNDDM         PUBLIC-INFORMATION,TYPE=PUB
STNFILE        TESTFDIC,DBID=123,FNR=007
```

When a user attempts to access the DDM PAYROLL-SALARY from the PAYLIB library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.TESTFDIC.DDM.PAYLIB.PAYROLL-SALARY
```

When the user attempts to access the DDM PUBLIC-INFORMATION, the DSN will not be generated because the STNDDM parameter TYPE=PUB. The FDIC is used to make the distinction between test and production versions of the DDM the user is accessing.

In order to operate more efficiently, SECURITRE maintains the full name of the last two DDMs successfully accessed by a user and bypasses DDM Security checking if the current DDM name matches one of the two saved names. These fields are cleared when the user logs on to a new library.

For example, assume the following parameters:

```
                                                                    Col. 72
STNPARM        PREFIX='NAT',QUALIFY='PROD',DDMMODE=FAIL,               x
               DDMLIT=DDM,DDMORDR=(LIT,LIB,DDM)
STNDDM         PAYROLL-SALARIES,ALIAS=SALARY,TYPE=PRIV
```

If the user tries to compile a program accessing the PAYROLL-SALARIES DDM while logged onto the library PAYLIB, SECURITRE will generate the following DSN:

```
NAT.PROD.DDM.PAYLIB.SALARY
```

SECURITRE then asks the SSF whether the user is authorized for this dataset. If the request is denied, the user will not be able to view or use the DDM while DDMMODE=FAIL.

If DDMORDR=DDM is specified, the following DSN will be generated:

```
NAT.PROD.SALARY
```

III.7.1 Relationship of DDM Security to Logon Security

The mode for DDM Security is set globally, not at the library level, by setting the DDMMODE parameter. However, since one of the components of the DDMORDR parameter is LIBrary name, security may be set up in such a way that a DDM may only be accessed within certain libraries.

For example, consider the following parameters:

```
                                                                    Col. 72
STNPARM         PREFIX='NATURAL',                                      x
                QUALIFY='',                                            x
                DELIM='.',                                             x
                DDMLIT=DDM,                                            x
                DDMMODE=FAIL,                                          x
                DDMORDR=(FDIC,LIT,LIB,DDM)
STNDDM          PAYROLL-SALARIES,ALIAS=PAYSAL,TYPE=PRIV
STNDDM          PAYROLL-EMPLOYEE,ALIAS=PAYEMP,TYPE=PRIV
```

Attempts to access the PAYROLL-SALARIES DDM and the PAYROLL-EMPLOYEES DDM from the library PAYLIB will generate the following DSNs:

```
NATURAL.DDM.PAYLIB.PAYSAL
NATURAL.DDM.PAYLIB.PAYEMP
```

As a result, access to these DDMs and all other DDMs with names beginning with "PAY" could be restricted to the PAYLIB library by granting authorized users access to the following DSN:

```
NATURAL.DDM.PAYLIB.PAY*
```

Attempts to access these DDMs from any other library will cause a change to the "PAYLIB" portion of the DSN, thus preventing the user from accessing the DDM from that library.

III.7.2 Relationship to SECURITRE ADABAS File Security

DDM Security can be used in conjunction with SECURITRE File Security to protect against unauthorized use of data. With the exception of RUN, DDM Security is not checked at execution time because at that point the concept of the DDM does not exist. However, when execution takes place, the data is secured through the SECURITRE User-Exit-1 (STRUEX1).

It is possible for a user to have access to a DDM, but be unable to access any data. For example, a user with access to the PAYROLL-SALARIES DDM might not have been granted access to the PAYROLL file itself. As a result, the user may RUN the program using the PAYROLL-SALARIES DDM and be unable to access any data.

III.8 Program Security

A user may access NATURAL programs in five different ways:

- READ program source (with a READ or RUN command)
- READ program object (EXECute the program)
- WRITE program source (with a SAVE or STOW command)
- WRITE program object (with a CAT or STOW command)
- DELETE program source or object (with a SCRATCH or PURGE command)

In order to control each of these types of access, SECURITRE offers several parameters related to program security. The Program Security parameters include PGLITOR, PGLITSR, PGLITOW, PGLITSW, PGLITPD, PGMORDR, and PGMTBSZ, which are specified in the STNPARM statement. Other parameters include PGMCHK and PGMTYPE, which are specified in the STNLIB statement.

If the PGMCHK parameter is set to WARN or FAIL, SECURITRE will generate a DSN when one of these types of accesses is detected and will check with the SSF to see if the user has authorization to the DSN. Violations will be logged by the SSF and if PGMCHK=FAIL, the user will not be allowed to access the program.

Example 1:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM     PGMTBSZ=30,PGLITOR=EXEC,PGLITSR=READ,PGLITPD=DEL,          x
            PGLITOW=CAT,PGLITSW=SAVE,PGMORDR=(LIB,LIT,PGM),            x
            PREFIX='NATURAL',QUALIFY='PROD',DELIM='.'
STNLIB      PAYROLL,PGMCHK=FAIL
STNFILE     PROD,DBID=1000,FNR=101
```

When a user attempts to read the PAYCHECK program from the Production PAYROLL library, SECURITRE will generate and check authorization to the DSN below regardless of the library the user is currently logged on to. (The user may have executed the command READ PAYCHECK PAYLIB while logged on to the PUBLIC library.)

```
NATURAL.PROD.PAYROLL.READ.PAYCHECK
```

Likewise, when users attempt to save their own PAYRAISE program in the PAYROLL library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PROD.PAYROLL.SAVE.PAYRAISE
```

Example 2:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM     PGMTBSZ=30,PGLITOR=READ,PGLITSR=READ,                      x
            PGLITOW=WRITE,PGLITSW=WRITE,                               x
            PGMORDR=(LIT,FUSER,LIB,PGM),                               x
            PREFIX='NATURAL',QUALIFY='',DELIM='.'
STNLIB      PAYROLL,PGMCHK=FAIL,PGMTYPE=PROG
STNFILE     PROD,DBID=1000,FNR=101
```

When a user executes the PAYCHECK program in the Production PAYROLL library, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.READ.PROD.PAYROLL.PAYCHECK
```

If the user attempts to save or catalog the PAYCHECK program, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.WRITE.PROD.PAYROLL.PAYCHECK
```

III.9 Relationship of Program Security to Logon Security

The mode for Program Security may be set at the library level by setting the PGMCHK parameter. For example, the site may specify that Program Security should be in FAIL mode for the PAYROLL library, while it is in DORMANT mode in all private libraries.

III.10 Run Security

RUN Security is used to prevent unauthorized users from executing NATURAL programs from source code. The mode for RUN Security is set at the library level, so that a site may guard against a RUN that is executed in some libraries but not others.

The RUN Security parameters include RUNORDR and RUNLIT, which are specified in the STNPARM statement, and RUNCHK, which is specified in the STNLIB statement.

When a RUN command is issued and the RUNCHK parameter is set to WARN or FAIL, SECURITRE will generate a DSN according to the RUNORDR and RUNLIT parameters and will check with the SSF to see if the user has authorization for the RUN. If the RUN is allowed, a flag is set in the user's buffer area to indicate that the user may issue a RUN from this library, so that no further checks on RUN Security will take place while the user is logged on to this library. This flag is turned off when the user logs on to a new library.

At the current time, no distinction is made between the RUN command and the RUN statement.

Since a user may bring any program source into the work area and execute a RUN with it, the program name is not always meaningful when a user RUNs a program. Therefore, when the DSN is generated for RUN Security, the program name will not be included in the RUNORDR parameter for generating the DSN.

Relationship of RUN Security to Logon Security

The mode for RUN Security may also be set at the library level by setting the RUNCHK parameter.

III.11 Other LOGON Security Controls

In addition to being used to control LOGON, PROGRAM, and RUN Security, the STNLIB parameters may be used to override default NATURAL settings at LOGON time. These parameters will be used even if a library is not secured by SECURITRE and will remain in effect until another LOGON is executed. These library parameters include STARTUP, ERRORTA, RDONLY, USRMODE, PGMWRT, LT, MT, MADIO, MAXCL, PGMCHK, and RUNCHK. They are defined in the STNLIB statement.

Example 1:
Assume the following parameters:

```
                                                                    Col. 72
STNLIB      PAYROLL,LIBFUSR=PROD,STARTUP=MENU,ERRORTA=ERRTA,          x
            USRMODE=NO,MADIO=500,MAXCL=50
STNLIB      PAYROLL,LIBFUSR=TEST,RDONLY=NO,USRMODE=YES
```

When users log on to the production PAYROLL library, they will immediately execute the "MENU" program. The "ERRTA" program will handle any errors that they may encounter. While logged on to the production PAYROLL library, they will not be able to issue NATURAL commands, the MADIO and MAXCL NATPARMS will be overridden, and the default LT and MT will be used. However, if users log on to the Test PAYROLL library, they will be able to issue NATURAL commands, and the default NATPARMS for LT, MT, MADIO, and MAXCL will be used.

Example 2:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM         RUNLIT=RUN,RUNORDR=(LIB,LIT),                        x
                PREFIX='NATURAL',QUALIFY='',DELIM='.'
STNLIB          PAYROLL,RUNCHK=FAIL
STNFILE         PROD,DBID=1000,FNR=101
```

Assume that a user obtains source code to a program that reads salary information. With Program Security in effect, the user is not allowed to execute this program. However, when the user logs on to the Production PAYROLL library, pulls the program into the work area, and executes a RUN, SECURITRE will generate and check authorization to the following DSN:

```
NATURAL.PAYROLL.RUN
```

If the user is not authorized for this DSN, the user will not be able to RUN the program.

Example 3:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM         RUNLIT=RUN,RUNORDR=(LIT,LIB,FUSER),                  x
                PREFIX='NATURAL',QUALIFY='PROD',DELIM='.'
STNLIB          PUBLIC,RUNCHK=WARN
STNFILE         PROD,DBID=1000,FNR=101
```

Assume that while a user is logged on to the PAYROLL library, they READ a program that they are not normally allowed to execute. Then, the user logs on to their private library, saves it under a name of a program to which they have access and executes a RUN. The following DSN will be generated:

```
NATURAL.PROD.RUN.JOEUSER.PROD
```

Since RUN Security is in WARN mode, the user manages to execute the program, but the attempt is logged on the system security log.

III.12 Fine Tuning

After SECURITRE for NATURAL is running, adjustments may be made to the parameters in order to fine-tune the security requirements. The following sections discuss the components of SECURITRE for NATURAL and how they may be combined to meet the site's security requirements.

STNFILE Statement

The STNFILE statement gives a meaningful name to a file and Database-ID combination. This name is used as an optional component of the various DSNs generated by SECURITRE for NATURAL. Several different files may use the same name.

Example 1:

```
                                                                  Col. 72
STNPARM        PREFIX='NATURAL',                                    x
               QUALIFY='',                                          x
               DELIM='.',                                           x
               NSIMODE=FAIL,                                        x
               NSIFDIC=NSI,                                         x
               NSIFNAT=NSI,                                         x
               NSIFUSR=NSI,                                         x
               NSIORDR=(LIT,FILE)
STNFILE        PROD,DBID=221,FNR=101
STNFILE        PROD,DBID=221,FNR=100
STNFILE        PROD,DBID=221,FNR=58
```

SECURITRE will generate and check authorization to the following DSN for all three NATURAL system file checks:

```
NATURAL.NSI.PROD
```

Given the parameters above, if NSIORDR=(FILE,LIT) is specified, SECURITRE will generate and check the following DSN:

```
NATURAL.PROD.NSI
```

If NSIMODE=DORMANT, SECURITRE will not generate a DSN and will ignore the NSIORDR, NSIFDIC, NSIFNAT, and NSIFUSR parameters. However, the STNFILE parameters may still be necessary if an FDIC or FUSER appears in any of the other SECURITRE for NATURAL order parameters.

The following statements might be used if the site wished to make the distinction between the FDIC, FNAT, and FUSR files on the different databases:

Example 2:
Assume the following parameters:

```
                                                                    Col. 72
STNPARM        PREFIX='NATURAL',                                      x
               QUALIFY='PROD',                                        x
               DELIM='.',                                             x
               NSIMODE=FAIL,                                          x
               NSIFDIC=NSIFDIC,                                       x
               NSIFNAT=NSIFNAT,                                       x
               NSIFUSR=NSIFUSR,                                       x
               NSIORDR=(FILE,LIT)
STNFILE        PRODFDIC,DBID=221,FNR=101
STNFILE        PRODFNAT,DBID=221,FNR=100
STNFILE        PRODFUSR,DBID=221,FNR=58
```

When a user attempts to invoke NATURAL with the NATPARMS FDIC=(221,101), FNAT=(221,100), and FUSER=(221,58), SECURITRE will generate the following DSNs:

```
NATURAL.PROD.PRODFDIC.NSIFDIC
NATURAL.PROD.PRODFNAT.NSIFNAT
NATURAL.PROD.PRODFUSR.NSIFUSR
```

If the user is not authorized for one or more of these DSNs, that user will not be given access to the NATURAL environment.

If a STNFILE statement has not been provided for a DBID/FNR combination, SECURITRE will generate a file name in the form DnnnFyyy, where 'nnn' refers to the Database-ID and 'yyy' refers to the file number (e.g., D115F083) if both of the values are less than 256, or else will generate a file name in the form Dnnnnn.Fyyyyy, where 'nnnnn' refers to the Database-ID and 'yyyyy' refers to the file number (e.g. D00275F01010).

Depending on the parameter values, up to three DSNs may be generated for NSI Security. One DSN will be used to check if the user has access to NATURAL using the FDIC specified in that user's NATPARMs; the second will be used to check access using the FNAT; and the third will be used to check access using the FUSER.

PGMTBSZ
If this parameter is increased, the size of the NATURAL buffer being used for a user will need to be increased as well. SECURITRE uses 12 bytes for each program in the table. If the user is given 1K of buffer storage, SECURITRE can maintain 20 programs in the program table. If the table is full and a program needs to be added to the table, SECURITRE will drop the oldest program out of the table and insert the new program.

The table is cleared out when the user logs on to a new library.

III.13 STNPARM Statements Sample

The sample statements below will set SECURITRE to function in an ADABAS/NATURAL environment.

```
                                                                      Col. 72
*
*
         STNPARM PREFIX=NATURAL                                          x
                 QUALIFY=TEST,                                           x
                 DELIM='.',                                              x
                 SERVER=246,                                             x
                 NSIMODE=WARN,                                           x
                 NSIORDR=(LIT,FILE),                                     x
                 NSIFNAT=NSIFNAT,                                        x
                 NSIFDIC=NSIFDIC,                                        x
                 NSIFUSER=NSIFUSER,                                      x
                 LGNMODE=WARN,                                           x
                 LGNLIT=LOGON,                                           x
                 LGNORDR=(LIT,LIB),                                      x
                 LGNPRIV=UID,                                            x
                 DDMMODE=WARN,                                           x
                 DDMLIT=DDM,                                             x
                 DDMORDR=(LIT,DDM),                                      x
                 PGLITOR=EXEC,                                           x
                 PGLITSR=RD,                                             x
                 PGLITOW=CAT,                                            x
                 PGLITSW=SAVE,                                           x
                 PGMORDR=(LIT,LIB,PGM),                                  x
                 PGMTBSZ=20,                                             x
                 RUNLIT=RUN,                                             x
                 RUNORDR=(LIT,LIB)
*
*       DEFAULT LIBRARY PARMS FOR PROD FUSER
*
         STNLIB *DEFAULT,LIBFUSR=PROD,TYPE=PRIV,PGMCHK=FAIL,USRMODE=NO,  x
                PGMTYPE=PROG
*
*       DEFAULT LIBRARY PARMS FOR ALL OTHER FUSERS
*
           STNLIB *DEFAULT,TYPE=PUB,PGMCHK=DORMANT,USRMODE=YES,          x
                RUNCHK=DORMANT
*
           STNLIB SYSTEM,TYPE=PUB,PGMCHK=D
           STNLIB SYSLIB,TYPE=PUB,PGMCHK=D
*
*       LIBRARY PARMS FOR PAYROLL IN PRODUCTION
*
STNLIB PAYROLL,LIBUSR=PROD,PGMCHK=FAIL,PGMTYPE=(PROG,MAP),              x
                USRMODE=NO,PGMWRT=NO,MT=100,LT=400,MADIO=300,MAXCL=100, x
                RUNMODE=FAIL,STARTUP=PAYMENU,ERRORTA=PAYERR
*
*       LIBRARY PARMS FOR PAYROLL LIBRARIES, NON-PRODUCTION
*
         STNLIB PAYROLL,PGMCHK=FAIL,PGMTYPE=PROG,RUNCHK=FAIL
*
*       DDM PARMS
*
           STNDDM *DEFAULT,TYPE=PUB
           STNDDM PAYROLL-PERSONNEL,ALIAS=PAYPERS,TYPE=PRIV
           STNDDM PAYROLL-PAYRATE,ALIAS=PAYRATE,TYPE=PRIV
*
*       FILE PARMS
*
           STNFILE PROD,DBID=100,FNR=001
           STNFILE PRODFDIC,DBID=100,FNR=002
           STNFILE STAGE,DBID=101,FNR=001
           STNFILE STAGEFD,DBID=101,FNR=1125
           STNFILE TEST,DBID=900,FNR=001
```

(continued on next page)

(continued from previous page)

```
*
       STNFINI
       END
```

Figure 7 – STNPARM Statements Sample

III.14 STNPARM Examples

This section provides examples of complete STNPARM groups. Each example shows how SECURITRE for NATURAL parameters interact with each other.

Example 1:

This is an example of a site that uses the same NATURAL module for all systems, including Test and Production. The site maintains a medium level of security. Assume the following parameters:

```
                                                                        Col. 72
STNPARM     DELIM='.',PREFIX='NAT',QUALIFY='',                            x
            DDMLIT=DDM,DDMMODE=FAIL,DDMORDR=(LIB,LIT,DDM),                x
            LGNLIT=LGN,LGNMODE=FAIL,LGNORDR=(LIB,LIT,FUSER),              x
            LGNPRIV=UID,                                                  x
            NSIFDIC=NSI,NSIFNAT=NSI,NSIFUSR=NSI,NSIMODE=FAIL,             x
            NSIORDR=(LIT,FILE),                                           x
            PGLITOR=EXEC,PGLITOW=PGMACC,PGLITSR=PGMACC,                   x
            PGLITSW=PGMACC,PGMORDR=(LIB,LIT,PGM,FUSER),                   x
            PGMTBSZ=20,
            PGWLIT=PWRITE,
            PGWORDR=(LIT,LIB,FUSER),
            RUNLIT=RUN,RUNORDR=(LIB,LIT,FUSER),                           x
            SERVER=179
*
STNLIB      *DEFAULT,LIBFUSR=PROD,TYPE=PRIV,PGMCHK=DORMANT,               x
            RUNCHK=DORMANT
*
STNLIB      *DEFAULT,TYPE=PUB,PGMCHK=DORMANT,RUNCHK=DORMANT
*
STNLIB      PAYROLL,LIBFUSR=PROD,                                         x
            STARTUP=MENU,ERRORTA=ERRTA,TYPE=PRIV,                         x
            PGMCHK=FAIL,PGMTYPE=(PROG,MAP),RUNCHK=FAIL,                   x
            USRMODE=NO,LT=500,MADIO=500
*
STNLIB      PAYROLL,LIBFUSR=TEST,TYPE=PRIV,PGMCHK=FAIL,                   x
            PGMTYPE=PROG,RUNCHK=FAIL
*
STNLIB      PAYROLL,TYPE=PRIV,PGMCHK=FAIL,PGMTYPE=PROG,                   x
            RUNCHK=FAIL,USRMODE=NO,PGWRTCK=FAIL
*
STNLIB      PUBLIC,LIBFUSR=PROD,STARTUP=MENU,ERRORTA=ERRTA,               x
            TYPE=PUB,USRMODE=NO,RDONLY=YES,PGMCHK=DORMANT,                x
            RUNCHK=DORMANT
*
STNLIB      PUBLIC,STARTUP=MENU,ERRORTA=ERRTA,TYPE=PUB,                   x
            USRMODE=YES,RDONLY=NO,PGMCHK=DORMANT,                         x
            RUNCHK=DORMANT
*
STNDDM      *DEFAULT,ALIAS=OTHER,TYPE=PRIV
STNDDM      PAYROLL-SALARY,ALIAS=SALARY,TYPE=PRIV
STNDDM      PAYROLL-PERSONNEL,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PAYROLL-PAYSCALES,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PAYROLL-MISCELLANEOUS,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PUBLIC-INFORMATION,TYPE=PUB
*
STNFILE     PROD,DBID=123,FNR=101          <- PRODUCTION FUSER
STNFILE     PROD,DBID=123,FNR=102          <- PRODUCTION FNAT
STNFILE     PROD,DBID=123,FNR=103          <- PRODUCTION FDIC
STNFILE     STG,DBID=135,FNR=1001          <- STAGE FUSER
STNFILE     STG,DBID=135,FNR=1002          <- STAGE FNAT
STNFILE     STG,DBID=135,FNR=1003          <- STAGE FDIC
STNFILE     TEST,DBID=157,FNR=101          <- TEST FUSER
STNFILE     TEST,DBID=157,FNR=102          <- TEST FNAT
STNFILE     TEST,DBID=157,FNR=103          <- TEST FDIC
STNFILE     DBA,DBID=200,FNR=101           <- DBA TEST FUSER
STNFILE     DBA,DBID=200,FNR=102           <- DBA TEST FNAT
STNFILE     DBA,DBID=200,FNR=103           <- DBA TEST FDIC
```

Assume that this site has the following users:

- USERA is a data-entry clerk in the Payroll department.
- USERB is a junior programmer in the MIS department.
- USERC is a senior programmer-analyst in the MIS department.
- USERD is a DBA.

Assume the following applications:

- PAYROLL is an application to process the site's payroll.
- PUBLIC is public information open to all users.

Information regarding all users:

SECURITRE will generate all DSNs beginning with the prefix "NAT", and use a delimiter character of "." (period) between all DSN components. All calls will be directed to User-Exit-1 on database 179, which will look up the user/DSN in its tables, and if not found, query the SSF to determine whether the user has access.

USERA - A Data-entry Clerk

When USERA first enters NATURAL, this user's dynamic NATPARMs include FNAT=(123,102), FDIC=(123,103), and FUSER=(123,101). SECURITRE generates the following DSN according to the instructions in NSIORDR:

```
NAT.NSI.PROD
```

Since the literals are the same for FNAT, FDIC, and FUSER, and the file name aliases are the same for all three files, only one DSN is generated. USERA has access to this DSN, so this user is allowed to enter NATURAL.

USERA's NATURAL parameters cause the user to immediately log on to the library PAYROLL. The LOGON causes SECURITRE to generate and check authorization for the following DSN:

```
NAT.PAYROLL.LGN.PROD
```

The user is authorized for this DSN and is allowed to log on to PAYROLL. Next, SECURITRE checks program write security. It generates the following DSN:

```
NAT.PAYROLL.PWRITE.PROD
```

This DSN is then sent to the SSF. The user is not authorized for this DSN, so the user will not be allowed to SAVE/CAT/STOW/PURGE any programs.

The startup transaction "MENU" is invoked (obtained from the STNLIB parameters PAYROLL for FUSER=PROD). At this time, SECURITRE sets NC=ON because USRMODE=NO, and it also overrides the NATPARM values for LT and MADIO.

USERA selects the PAYUPDT program, and SECURITRE generates and checks the following DSN for which USERA has authorization:

```
NAT.PAYROLL.EXEC.PAYUPDT.PROD
```

PAYUPDT calls the PAYUPD1 map, so SECURITRE calls the SSF to see if the user may access the following DSN for which USERA is allowed access:

```
NAT.PAYROLL.EXEC.PAYUPD1.PROD
```

PAYUPDT also calls the PAYSUB subroutine, but SECURITRE does not do a security check because PGMTYPE does not include SUBR.

USERA selects a menu item that logs this user on to the PUBLIC application. No security checks are made because the TYPE for library PUBLIC=PUB. Again, the startup transaction MENU is invoked. USERA may not enter any NATURAL commands or update any ADABAS files here because of the USRMODE and RDONLY parameters. However, the user may execute any programs in this library without security checking.

USERB - A Junior Programmer

When USERB first enters NATURAL, this user's dynamic NATURAL parameters include FNAT=(157,102), FDIC=(157,103), and FUSER=(157,101). SECURITRE will generate and check authorization for the following DSN:

```
NAT.NSI.TEST
```

USERB is allowed to enter NATURAL with those parameters and is logged on to USERB's private library, USERB. SECURITRE does not check LOGON Security for this library because the parameter LGNPRIV=UID. If the user was set up to log on to USERBLIB, SECURITRE will generate and check the following DSN:

```
NAT.USERBLIB.LGN.TEST
```

The default settings for parameters, including USRMODE, RDONLY, LT, and MT, are taken because SECURITRE has chosen the second set of *DEFAULT STNLIB parameters, and these parameters are not overridden.

USERB may read and write any program in USERB's library. However, when USERB tries to read the program "PAYUPDT" from the PAYROLL library while still logged on to the USERB library, USERB issues the command "READ PAYUPDT PAYROLL". SECURITRE will generate and check the DSN below because NATURAL is obtaining PAYUPDT from PAYROLL, a library where PGMCHK is in FAIL mode. USERB receives an error and is not able to read the program.

```
NAT.PAYROLL.PGMACC.PAYUPDT.TEST
```

USERB then attempts to write a PAYUPDT program. In the editor, USERB enters the following lines of code:

```
FIND PAYROLL-SALARY WITH LAST-NAME = 'USERB'
UPDATE WITH ANNUAL-SALARY = 50000
```

When NATURAL attempts to obtain the DDM PAYROLL-SALARY, SECURITRE generates and checks the following DSN, and USERB gets an error:

```
NAT.USERB.DDM.SALARY
```

USERB gives up the attempt to update the USERB salary and settles for looking at the PUBLIC-INFORMATION file instead. USERB writes a small program to read records from PUBLIC-INFORMATION. SECURITRE does not check DDM Security for USERB because the TYPE for PUBLIC-INFORMATION is set to PUB. When the user finishes entering the program, this user issues a RUN command. SECURITRE will generate and check the following DSN that the user has been authorized access to by the SSF:

```
NAT.USERB.RUN.TEST
```

USERB then attempts to enter the production system by using the NATURAL parameters that point to the production NATURAL files. SECURITRE will generate and check the following DSN and USERB is not allowed into NATURAL:

```
NAT.NSI.PROD
```

USERC - A Senior Systems Analyst

USERC has been given access to anything in the PAYROLL and PUBLIC-INFORMATION libraries, including programs and DDMs. This user may only execute PAYROLL update programs in TEST and STG. When USERC executes PAYUPDT in STG, for example, SECURITRE will generate and check the following DSN, and USERC is allowed to execute the program:

```
NAT.PAYROLL.EXEC.PAYUPDT.STG
```

However, when USERC attempts to execute PAYUPDT in PROD, SECURITRE will generate and check the following DSN, and USERC will receive an error because the DSN was rejected by the SSF:

```
NAT.PAYROLL.EXEC.PAYUPDT.PROD
```

USERC is not allowed to use the TEST DBA system files. If USERC attempts to access the DBA NATURAL, SECURITRE will reject the user because the SSF will reject USERC's access to the following DSN:

```
NAT.NSI.DBA
```

USERC must go through the same security checks as USERA and USERB. However, the SSF allows all authorization requests, except Production PAYROLL updates, and USERC is able to navigate through the system without being stopped by SECURITRE. Note that if USERC had written a program in Production to update a payroll file, SECURITRE File Security will stop USERC. When the user logs on to the PAYROLL library on STG, they are unable to enter any NATURAL commands and are restricted to the functionality provided by the user transactions because USRMODE=NO. (This is also true when the user logs on to the PAYROLL library on PROD.) The user really does not need the NATURAL commands because all development is done at the TEST level.

USERD - A DBA

USERD, the DBA, is allowed access to everything, including the DBA database. USERD changes the USERD password weekly and does not write it down. If necessary, USERD can perform a "privileged" logon to a library, and no security checks will be performed while the DBA is logged on to that library.

Example 2:
This is an example of a site that uses different NATURAL modules for each system. This particular set of parameters is used on the production system.

Assume the following parameters:

```
                                                                        Col. 72
STNPARM     DELIM='.',PREFIX='NAT',QUALIFY='PROD',                          x
            DDMLIT=DDM,DDMMODE=FAIL,DDMORDR=(LIT,LIB,DDM),                 x
            LGNLIT=LGN,LGNMODE=FAIL,LGNORDR=(LIT,LIB),                     x
            LGNPRIV=NONE,                                                  x
            NSIFDIC=NSIFDIC,NSIFNAT=NSIFNAT,NSIFUSR=NSIFUSR,               x
            NSIMODE=FAIL,NSIORDR=(LIT,FILE),                               x
            PGLITOR=EXEC,PGLITOW=CAT,                                      x
            PGLITSR=READ,PGLITSW=SAVE,PGMORDR=(LIT,LIB,PGM),               x
            PGMTBSZ=60,                                                    x
            RUNLIT=RUN,RUNORDR=(LIT,LIB),                                  x
            SERVER=200

STNLIB      *DEFAULT,TYPE=PRIV,PGMCHK=DORMANT,RUNCHK=DORMANT
*
STNLIB      PAYROLL,                                                       x
            STARTUP=MENU,ERRORTA=ERRTA,TYPE=PRIV,                          x
            PGMCHK=FAIL,PGMTYPE=(PROG,MAP),RUNCHK=FAIL,                    x
            USRMODE=NO,LT=500,MADIO=500
*
STNLIB      PUBLIC,STARTUP=MENU,ERRORTA=ERRTA,                             x
            TYPE=PUB,USRMODE=NO,RDONLY=YES,PGMCHK=DORMANT,                 x
            RUNCHK=DORMANT
*
STNDDM      *DEFAULT,ALIAS=OTHER,TYPE=PRIV
STNDDM      PAYROLL-SALARY,ALIAS=SALARY,TYPE=PRIV
STNDDM      PAYROLL-PERSONNEL,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PAYROLL-PAYSCALES,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PAYROLL-MISCELLANEOUS,ALIAS=PAYROLL,TYPE=PRIV
STNDDM      PUBLIC-INFORMATION,TYPE=PUB
*
STNFILE     PRODFDIC,DBID=123,FNR=103              <- PRODUCTION FDIC
STNFILE     PRODFNAT,DBID=123,FNR=102              <- PRODUCTION FNAT
STNFILE     PRODFUSR,DBID=123,FNR=101              <- PRODUCTION FUSER
```

Assume the site has the following applications:

- PAYROLL is an application to process the site's payroll.
- PUBLIC is public information open to all users.

Information regarding all users:

SECURITRE will generate all DSNs beginning with the prefix and qualifier "NAT.PROD" and use a delimiter character of "." (period) between all DSN components. All calls will be directed to User-Exit-1 on database 123, which will look up the user/DSN in its tables. If User-Exit-1 does not find the user/DSN, it queries the SSF to determine whether or not the user has access.

NSI Security

NSI Security has been set up on this library so that each user will be checked for each system file. When a user attempts to enter this NATURAL with the NATPARMs FNAT=(123,102), FDIC=(123,103), and FUSER=(123,101), SECURITRE will generate the following DSNs:

```
NAT.PROD.NSIFDIC.PRODFDIC
NAT.PROD.NSIFNAT.PRODFNAT
NAT.PROD.NSIFUSR.PRODFUSR
```

If the user is authorized for these datasets, the user will be allowed into NATURAL. For example, if a user had tried to enter NATURAL with the NATPARMS FDIC=(157,103), FNAT=(157,102), and FUSER=(157,101), SECURITRE will generate and check authorization for the following DSNs:

```
NAT.PROD.NSIFDIC.D157F103
NAT.PROD.NSIFNAT.D157F102
NAT.PROD.NSIFUSR.D157F101
```

If the user had tried to enter NATURAL with the NATPARMS FDIC (157,303), FNAT=(157, 302), and FUSER=(157, 301), SECURITRE will generate and check authorization for the following DSNs:

```
NAT.PROD.NSIFDIC.D00157.F00303
NAT.PROD.NSIFNAT.D00157.F00302
NAT.PROD.NSIFUSR.D00157.F00301
```

LOGON Security

LOGON Security has been set up in this system, so SECURITRE will check LOGONs to all libraries except the PUBLIC library. Since STNPARM LGNPRIV=NONE, security checks will take place when a user attempts to logon to libraries that begin with that user's User-ID. For example, if a user attempts to logon to the PAYROLL library, SECURITRE will generate and check authorization for the following DSN:

```
NAT.PROD.LGN.PAYROLL
```

DDM Security

DDM Security will only be checked if the DDMs for the payroll files are referenced, either from within a source program or through a 'LIST FILE'. DDM checking will not occur on the PUBLIC-INFORMATION DDM. For example, if a user accesses the PAYROLL-PERSONNEL DDM while logged on to the PAYROLL library, SECURITRE will generate and check authorization to the following DSN:

```
NAT.PROD.DDM.PAYROLL.PAYROLL
```

But if the user accesses PAYROLL-PERSONNEL while logged on to the PUBLIC library, SECURITRE will generate and check authorization to the following DSN:

```
NAT.PROD.DDM.PUBLIC.PAYROLL
```

RUN Security

RUN Security will only be checked in the PAYROLL library, since RUNCHK=DORMANT everywhere else. If a RUN statement was included in a NATURAL program executed by a user in this library, SECURITRE will generate and check authorization to the following DSN:

```
NAT.PROD.RUN.PAYROLL
```

If access was granted, the user would not have to be checked again while logged on to this library, since SECURITRE "remembers" that RUN access was allowed from this library. However, if the user logs on to another library and then back to this library, SECURITRE will re-check the user's access.

PROGRAM Security

Program Security will only be checked in the PAYROLL library. If a user attempts to execute the PAYUPDT program, SECURITRE will generate and check authorization to the following DSN:

```
NAT.PROD.EXEC.PAYROLL.PAYUPDT
```

If a user attempts to read the source for the PAYUPDT program while logged on to the private library USERBLIB, SECURITRE will generate and check authorization to the following DSN because PAYUPDT is being obtained from PAYROLL:

```
NAT.PROD.READ.PAYROLL.PAYUPDT
```

And finally, if PAYUPDT calls the PAYHELP help routine, a DSN will not be generated because HELP is not specified in the PGMTYPE list.

For each user, SECURITRE will maintain a list of up to 60 programs that have been approved for execution while a user is logged on to a library. The list is purged when the user logs on to a new library.

Example 3:

This is an example of a site that uses different NATURAL modules for each system. This particular set of parameters is used on the test system.

Assume the following parameters:

```
                                                                        Col. 72
STNPARM     DELIM='.',PREFIX='NAT',QUALIFY='TEST',                        x
            DDMMODE=DORMANT,                                              x
            LGNLIT=LGN,LGNMODE=FAIL,LGNORDR=(LIT,LIB),                    x
            LGNPRIV=UID+,                                                 x
            NSIFDIC=NSI,NSIFNAT=NSI,NSIFUSR=NSI,                          x
            NSIMODE=FAIL,NSIORDR=(LIT,FILE),                              x
            SERVER=200
*
STNLIB      *DEFAULT,PGMCHK=DORMANT,RUNCHK=DORMANT,TYPE=PRIV
*
STNFILE     TESTFDIC,DBID=123,FNR=103               <- TEST FDIC
STNFILE     TESTFNAT,DBID=123,FNR=102               <- TEST FNAT
STNFILE     TESTFUSR,DBID=123,FNR=101               <- TEST FUSER
```

Information regarding all users:

SECURITRE will generate all DSNs beginning with the prefix and qualifier "NAT.TEST" and use a delimiter character of "." (period) between all DSN components. All calls will be directed to User-Exit-1 on database 123, which will look up the user/DSN in its tables. If User-Exit-1 does not find the user/DSN, it queries the SSF to determine whether or not the user has access.

NSI Security

NSI Security has been set up for this NATURAL, so the DSNs will all include the same literal, but not the same file name. When a user attempts to enter this NATURAL with the NATPARMs FDIC=(123,103), FNAT=(123,102), and FUSER=(123,101), SECURITRE will generate and check authorization to the following DSNs before allowing the user to enter NATURAL:

```
NAT.TEST.NSI.TESTFDIC
NAT.TEST.NSI.TESTFNAT
NAT.TEST.NSI.TESTFUSR
```

LOGON Security

LOGON Security is in effect for all libraries for this NATURAL, except for private libraries. For example, a user attempting to logon to a library that begins with their User-ID will not be checked. Suppose a user attempts to logon on to the PAYROLL library. SECURITRE will generate and check authorization to the following DSN:

```
NAT.TEST.LGN.PAYROLL
```

There is no Program or RUN Security in effect for this NATURAL module.

Example 4:

Assume the following parameters:

```
                                                                        Col. 72
STNPARM    DELIM='.',PREFIX='NAT',QUALIFY='PROD',DDMMODE=DORMANT,          x
           LGNLIT=LGN,LGNMODE=FAIL,LGNORDR=(LIT,LIB),                     x
           LGNPRIV=UID+,SERVER=123,                                       x
           NUMODE=FAIL,NULIT=UTIL,NUORDR=(LIT,LIB,UTIL)
*
STNLIB     *DEFAULT,PGMCHK=DORMANT,RUNCHK=DORMANT,TYPE=PRIV
STNLIB     PAYROLL,TYPE=PRIV,PGMCHK=DORMANT,RUNCHK=DORMANT,               x
           STEP1=PAY1,STEP2=PAY2,STEP3=PAY3,STEP4=SUBPLIB                 x
           STEP5=SYSTEM

STNFILE    PRODFDIC,DBID=123,FNR=103
STNFILE    PRODFNAT,DBID=123,FNR=102
STNFILE    PRODFUSR,DBID=123,FNR=101
```

Information regarding all users:

SECURITRE will generate all DSNs beginning with the prefix and qualifier "NAT.PROD" and will use a delimiter character of "." (period) between all DSN components. All calls will be directed to User-Exit-1 on database 123, which will look up the user/DSN in its tables. If User-Exit-1 does not find the user/DSN, it queries the SSF to determine whether or not the user has access.

Information pertaining to each user will be stored in an allocated area for the user. The modules STNGET and STNFREE, which perform the allocate and FREEMAIN for the buffer space, are linked in the NATURAL nucleus.

LOGON Security

LOGON Security will be checked for all libraries in this NATURAL nucleus, except for those libraries that begin with a user's own User-ID. For example, if a user attempts to log on to the PAYROLL library, SECURITRE will generate and check authorization to the following DSN:

```
NAT.PROD.LGN.PAYROLL
```

STEPLIBS

If the user is allowed to log on to the PAYROLL library, SECURITRE will assign the libraries named PAY1, PAY2, PAY3, SUBPLIB, and SYSTEM as the steplibs for this library. That is, if a particular NATURAL object is called and it is not found in PAYROLL, NATURAL will search for it in PAY1, PAY2, PAY3, SUBPLIB, and SYSTEM.

NATURAL Utility Security

SECURITRE will be used to secure all NATURAL utilities in this example. If a user is logged on to the library ABCLIB and tries to execute SYSMAIN, SECURITRE will generate and check authorization for the DSN:

```
NAT.PROD.UTIL.ABCLIB.SYSMAIN
```

SECTION IV

SECURITRE FOR ADABAS UTILITIES

IV.1 Introduction

SECURITRE makes it possible to secure the ADABAS utilities, which is an important component of a total security solution. Controlling the reading and writing of individual ADABAS records through the ADABAS Nucleus does not completely secure ADABAS files from modification. Anyone with authorization to execute ADABAS utilities can erase an entire file or make other significant changes to the database because some utilities do not go through the ADABAS Nucleus.

Using the SECURITRE for ADABAS Utilities feature, security can be established to restrict anyone who may run a utility or any of its functions. Security on utilities may be further tightened to restrict the database or file against which a user may run each utility and function.

This section describes the flow of utility execution when SECURITRE is not installed and shows how easily SECURITRE can be installed to secure ADABAS utilities.

For additional information related to utility security, please review the Setting Up SECURITRE for ADABAS section and the Installation section of this manual.

The Setting Up SECURITRE for ADABAS section of this manual details the STRPARM module parameters, which allow users to customize SECURITRE at their site with some of the parameters applying to Utility Security.

The Installation section of this manual provides installation instructions for SECURITRE, including instructions for installing the necessary components to secure ADABAS utilities.

Utility Execution Without Securitre

One of the functions of the ADARUN module supplied with ADABAS is to "direct" utility execution. As shown in the figure on the following page, ADARUN reads DDCARD input to validate the parameters and decide which utility to run. Then, control is passed to the requested utility program.

The utility reads DDKARTE parameters specific to the utility and determines which function to perform and which file to use. Then the utility executes, performing the requested task, frequently involving interaction with the database. However, not all utilities communicate with the ADABAS Nucleus.

By itself, ADARUN does not provide a mechanism to control which utilities are allowed to be executed or which users are allowed to execute them. By themselves, the SSFs can control who may execute the program ADARUN, but it is an all or nothing proposition. In other words, when given the authority to run ADARUN, the SSF will allow any ADABAS utility to be run against any file, which is clearly unacceptable in any security-conscious ADABAS site.

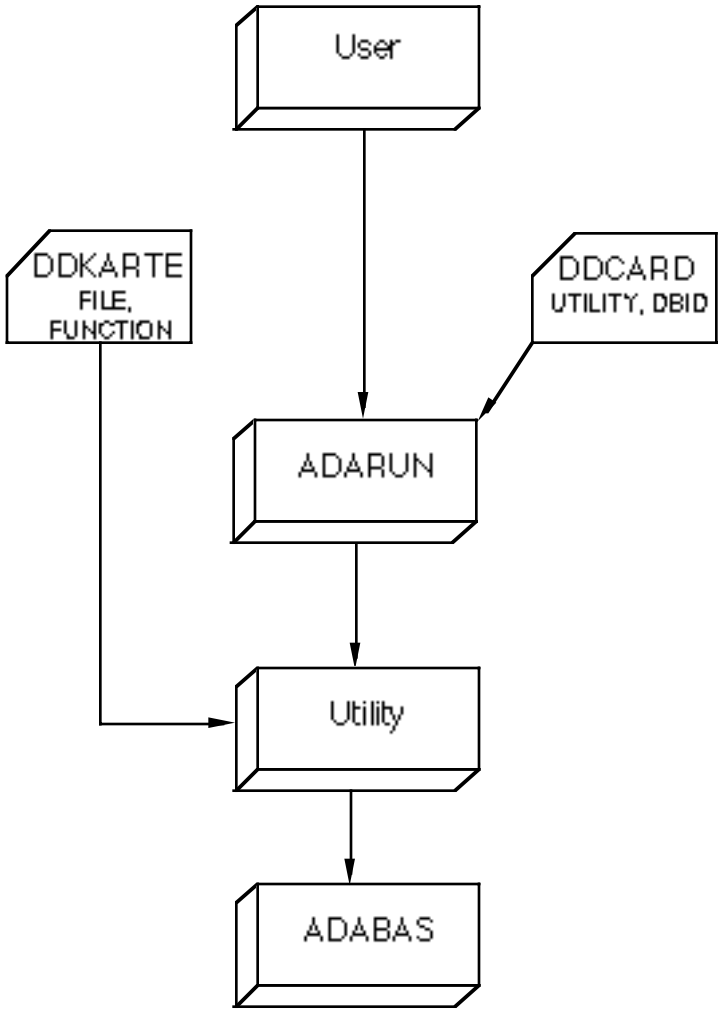

Figure 8 – Utility Execution Without SECURITRE

Utility Execution With Securitre

SECURITRE uses a front-end that is link-edited to the ADARUN module, which allows it to check security for a user before ADARUN is permitted to execute. The function of the SECURITRE front-end is to generate an appropriate DSN and call the SSF to verify whether or not this particular user is allowed to execute this utility (access the DSN).

The SECURITRE front-end reads both DDCARD and DDKARTE input to determine the utility and function to be performed, as well as the database and file to be used (refer to the following figure). In addition, the front-end uses the STRPARM module generated by the STRDEF and STRFNR macros. The DSN generated by SECURITRE is based on the parameters that are specified by these macros, combined with the utility requested in the DDCARD input. Where applicable, the file and function are included in the DSN.

After SECURITRE has generated a DSN, a call is made to the SSF to determine whether or not the user may perform the requested utility and function against the specified database and file. If the user has the appropriate authority, control will be passed to the ADARUN module and utility processing will continue as if SECURITRE was not involved. If the user fails the security check, the utility will be ABENDed and no further processing will occur.

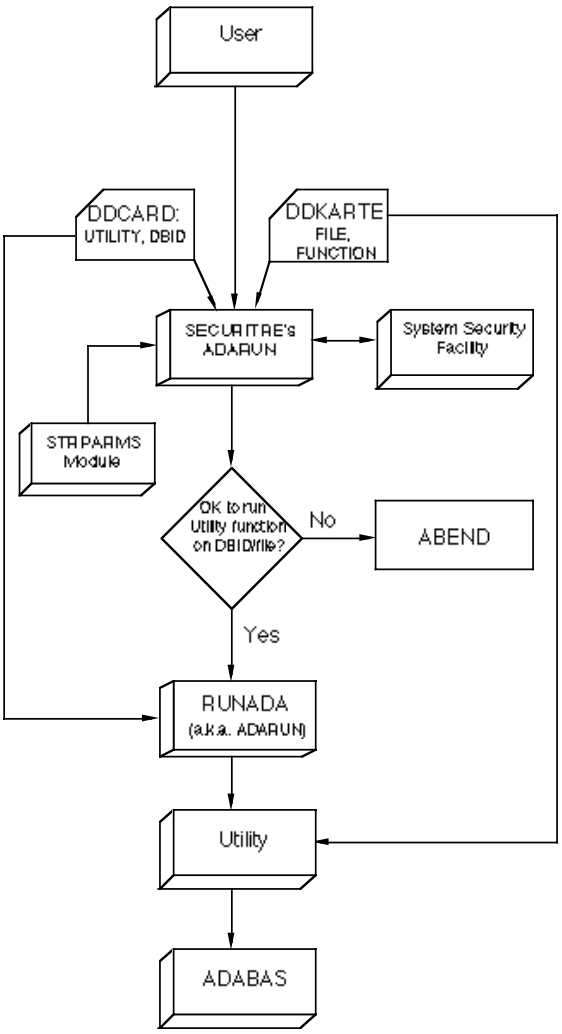

Figure 9 – Utility Execution With SECURITRE

IV.2 Relationship of Utility Security to File Security

Utility Security checking takes place before ADARUN is invoked. Once a user has been approved, control is passed to ADARUN. Then, the utility may send calls to ADABAS, where they can be reviewed by STRUEX1. Therefore, it is possible for a user to be allowed to run a utility through SECURITRE Utility Security and still be prevented from running against the database/file by User-Exit-1.

DSNs generated by the Utility Security front-end to ADARUN will not affect the User-Exit-1 DSNPOOL table and will not appear on the STRUEX1 trace unless the utility calls ADABAS. Any ADABAS calls initiated by the utility may result in an entry being added to the DSNPOOL table. ADAULD and ADAREP CPLIST are examples of utilities that issue ADABAS calls.

IV.3 Utility Security Entity Naming Conventions

All utility DSNs will be derived from the value of the UTPREF and UTORDER parameters set in the STRPARM module. For more information, refer to **Section II - Setting Up SECURITRE for ADABAS**.

The value of the UTPREF parameter will be used as the first segment, or "prefix", of the DSN. This prefix may be up to 17 characters long.

The remainder of the DSN will be determined by the order that the STRPARM parameter UTORDER indicates. There are three possible components to the UTORDER parameter: UTIL, FUNC and FILE. Any or all of these components may be specified, in any desired order.

If UTIL is included in the UTORDER parameter the last three characters of the ADABAS utility name will be included in the DSN.

When FUNC is included in the UTORDER parameter and there is a valid function specified on the DDKARTE input, the function will be included in the DSN. If the function name is longer than eight characters, it will be truncated to eight characters, as indicated in ***Section IV.2 – ADABAS V5, V6, and V7 Utility Control*** in the ***SECURITRE Reference Manual***.

FILE may also be included in the UTORDER parameter. When there is a file number contained in the DDKARTE input, the file name from the STRDEF or STRFNR parameters will be included in the DSN. For more information, refer to ***Section IV.2 – ADABAS V5, V6, and V7 Utility Control*** in the ***SECURITRE Reference Manual***.

The following examples show how DSNs may be generated for SECURITRE Utility Security. For all these examples, assume that the STRDEF UTPREF parameter is set to:

```
UTPREF='ADABAS.UTL'
```

Also, assume that no STRFNR alias file name overrides exist.

Example 1:
If UTORDER=(UTIL,FUNC,FILE), the DDCARD contains ADARUN PROG=ADASAV, and the DDKARTE contains ADASAV RESTONL FILES=1,4,266, SECURITRE will generate the following DSNs:

```
ADABAS.UTL.SAV.RESTONL.F001
ADABAS.UTL.SAV.RESTONL.F004
ADABAS.UTL.SAV.RESTONL.F00266
```

Example 2:
If UTORDER=(UTIL), the DDCARD contains ADARUN PROG=ADASAV, and the DDKARTE contains ADASAV RESTONL FILES=1,4,266, SECURITRE will generate the following DSN:

```
ADABAS.UTL.SAV
```

Example 3:
If UTORDER=(FILE,UTIL), the DDCARD contains ADARUN PROG=ADASAV, and the DDKARTE contains ADASAV RESTONL FILES=1,4,266, SECURITRE will generate the following DSNs:

```
ADABAS.UTL.F001.SAV
ADABAS.UTL.F004.SAV
ADABAS.UTL.F00266.SAV
```

Example 4:
Assume UTORDER=(UTIL,FUNC,FILE), the DDCARD contains ADARUN PROG=ADADBS, and the DDKARTE contains:

```
ADADBS OPERCOM DAUQ
ADADBS OPERCOM DFILES=1,4
ADADBS OPERCOM DFILUSE=266
ADADBS OPERCOM DLOCKF
```

SECURITRE will generate the following DSNs:

```
ADABAS.UTL.DBS.OPER.DAUQ
ADABAS.UTL.DBS.OPER.DFILES.F001
ADABAS.UTL.DBS.OPER.DFILES.F004
ADABAS.UTL.DBS.OPER.DFILUSE.F00266
ADABAS.UTL.DBS.OPER.DLOCKF
```

Example 5:
If UTORDER=(FILE,UTIL), the DDCARD contains ADARUN PROG=ADADBS, and the DDKARTE contains:

```
ADADBS OPERCOM DAUQ
ADADBS OPERCOM DFILES=1,4
ADADBS OPERCOM DFILUSE=266
ADADBS OPERCOM DLOCKF
```

SECURITRE will generate the following DSNs:

```
ADABAS.UTL.F001.DBS
ADABAS.UTL.F004.DBS
ADABAS.UTL.F00266.DBS
ADABAS.UTL.DBS
```

Example 6:
If UTORDER=(UTIL,FUNC,FILE), the DDCARD contains ADARUN PROG=ADADBS, and the DDKARTE contains:

```
ADADBS RELEASE field on File 3
ADADBS RESETDIB
ADANUC
```

SECURITRE will generate the following DSNs:

```
ADABAS.UTL.DBS.RELEASE.F003
ADABAS.UTL.DBS.RESETDIB
ADABAS.UTL.NUC
```

Example 7:
If UTORDER=(FILE,UTIL), the DDCARD contains ADARUN PROG=ADADBS, and the DDKARTE contains:

```
ADADBS RELEASE field on File 3
ADADBS RESETDIB
ADANUC
```

SECURITRE will generate the following DSNs:

```
ADABAS.UTL.F003.DBS
ADABAS.UTL.DBS
ADABAS.UTL.NUC
```

IV.4 Utility Security Mode

Utility Security may be run in DORMANT, WARN, or FAIL mode, according to the value assigned to the UTMODE parameter. The STRFNR MODE parameter is not used to override the UTMODE because confusion may arise when handling multiple files on an input DDKARTE or when the lack of files on a DDKARTE indicates that all files are to be processed during a utility run.

IV.5 Utility Run ABENDs and Error Messages

As part of the normal Utility Security checking process, SECURITRE interprets all of the utility run cards submitted for the utility run and determines the Database-ID, file number, ADABAS utility, and utility function specified. This information is then used to determine if a particular user is authorized to execute a utility. During this process, SECURITRE may detect errors or inconsistencies, or determine that a user is not authorized to execute the utility.

If SECURITRE detects an error condition or insufficient authorization, an error message about the condition encountered will be printed to the console and ABEND the utility run.

If an STRMSG DD card is included in the JCL, SECURITRE will print the contents of DDCARD and DDKARTE to the STRMSG dataset.

For more information about Utility Security ABEND Codes and Messages, refer to ***Section IV.3 - Utility Security Error Messages*** in the ***SECURITRE Reference Manual***.

SECTION V

REAL-TIME MONITOR

V.1 Introduction to the Real-Time Monitor

The SECURITRE Real-Time Monitor (RTM) provides the Security Administrator with a powerful tool that assists in the control of on-line ADABAS security. With the RTM, the Security Administrator may perform the following tasks:

- Remove a user from the SECURITRE tables, causing a refresh of that user's file access list.
- Remove all users from the SECURITRE tables, causing a refresh of the file access lists for all users.
- Display the current SECURITRE "STRPARM" parameter settings for ADABAS, such as the STRDEF and STRFNR settings.
- Display the current SECURITRE "STNPARM" parameter settings for NATURAL.
- Reload the SECURITRE "STRPARM" module into the ADABAS region. This allows any off-line modifications that have been made to the parameters to be implemented without bringing the database down and up.
- Reload the SECURITRE user-exits and any other ADABAS User-Exit-1 into the ADABAS region, allowing any off-line modifications to these exits to be implemented without bringing the database down and up.
- Alter the status of the SECURITRE Trace Facility, providing a diagnostic trace of activity within the SECURITRE ADABAS User-Exit-1 that may be activated or deactivated from the RTM.
- Invoke the TRIM RTM, if installed, allowing the transfer of control from the SECURITRE RTM to the TRIM RTM with a menu selection. The TRIM RTM also provides for the transfer of control to the SECURITRE RTM.

The RTM is very simple to install and use. In fact, a simple NATLOAD procedure may be used to perform the installation of all RTM NATURAL modules, and there are no Zaps required to any ADABAS modules, SVC, or Link Routines. The supplied RTM NATURAL Modules must be loaded on at least one database, preferably "Test". Each database to be monitored/controlled must have the SECURITRE User-Exit-1 and User-Exit-4 that are supplied. To use the RTM, an STRRTM and a RTMORDR parameter must be defined within the STRDEF statement.

No Command Log records need to be written to use this RTM.

SECURITRE user-exits can "co-exist" with the user's previously installed user-exits.

Once ADABAS is brought up with SECURITRE User-Exit-1 and User-Exit-4, the RTM is available for use.

If the product NATURAL SECURITY System (NSS) is being used, the application upon which the RTM modules are installed must be defined for use by all potential users of the RTM.

The SECURITRE RTM operates using any teleprocessing system that supports NATURAL and ADABAS.

For more information about installing the RTM, refer to the Installation section of this manual.

For more information about RTM functions, screen names, and a further explanation of usage, refer to ***Section V – Real-Time Monitor*** in the ***SECURITRE Reference Manual***.

V.2 RTM Security

SECURITRE can be used to secure itself. That is, the ability to execute the various SECURITRE RTM functions can be controlled through a SECURITRE request to the SSF.

Within the STRPARM parameter module, the STRDEF macro STRRTM parameter defines the high level prefix of a DSN. The RTMORDR parameter specifies whether or not the RTM function and/or the RTM DBID will be included in the DSN. When SECURITRE detects a request from the RTM, access to the appropriate DSN is automatically checked with the SSF. If access is allowed, the RTM request is processed. If access is denied, the RTM request is rejected.

For example, assume the STRDEF statement parameters:

```
                                                                  Col. 72
                                                                    x
STRRTM='SECURITRE.RTM',
RTMORDR=FUNC
```

Before the RTM user can cause a parameter reload function, "RPRM", the user validates the DSN below through an automatic call to the SSF by SECURITRE.

```
SECURITRE.RTM.LODPARM
```

For more information about the various screen names, their functions, and how those functions are represented in the DSN, refer to ***Section V.4 - RTM Security*** in the ***SECURITRE Reference Manual***.

V.3 RTM Operation

The SECURITRE RTM is a menu-driven set of NATURAL modules. The user simply logs onto whichever library the RTM modules have been installed on and executes MENU. There are no new languages or disciplines to learn because the RTM functions like any other ADABAS/NATURAL application. All RTM module names start with "STR" or "STN", except for the MENU program.

For more information about RTM functions and screen names, as well as a further explanation of usage, refer to ***Section V - Real-Time Monitor*** in the ***SECURITRE Reference Manual***.

Multiple Database Support

The NATURAL modules representing the RTM may be installed on one database but perform monitoring/control of others. It is common to install the RTM on one Test database while directing it to monitor/control several other Test and/or Production databases.

When the NATURAL modules are placed on the Test database, most overhead associated with displays and fetches of the RTM NATURAL modules will affect the Test database, but it will not affect the Production database even though the Production databases are being monitored/controlled.

The database that is being monitored/controlled is displayed on nearly every screen. However, the database to be monitored may also be changed directly on the top of nearly every detail screen. For example, the display of SECURITRE parameters in effect may be switched from DBID 123 to DBID 1023 by entering the number 1023 in the DBID field on the parameter display screen.

This page intentionally left blank.

SECTION VI

INTERNAL APPLICATION SECURITY FEATURES (STRNAT AND STRASM)

VI.1 Introduction

In applications there may be a need to provide various levels of security beyond database, file, and field levels, including:

- Date or time oriented security
- Function/screen level security
- Data protection on the field level beyond what is provided in SECURITRE for ADABAS
- Security by field value

Prior to SECURITRE, a "hard coded" approach using tables or Security files may have been used. For example, assume that a NATURAL payroll application is used to generate the paychecks for all employees in the organization. If access to payroll information is limited by DEPARTMENT, logic like the following could be "hard coded" to limit access to payroll data:

```
READ PAYROLL-DATA BY DEPARTMENT
AT BREAK OF DEPARTMENT
  FOR I=1 TO 200                          /* 200 USERS IN TABLE
    IF *USER = TABLE-USER(I)
       MOVE TRUE TO FOUND                 /* GOT A HIT
       FOR J = 1 TO 10                    /* 10 DEPARTMENTS
          IF DEPARTMENT = TAB-DEPT(I,J)
             MOVE TRUE TO SEC-OK          /* ACCESS IS ALLOWED
             ESCAPE BOTTOM
          END-IF
       END-FOR
       ESCAPE BOTTOM                      /* GET OUT
    END-IF
  END-FOR
  IF FOUND                                /* USER-ID FOUND?
IF SEC-OK                                 /* ACCESS ALLOWED?
       IGNORE
    ELSE
       MOVE TRUE TO EOF
       ESCAPE BOTTOM IMMEDIATE
    END-IF
  ELSE
    PERFORM S900-FATAL-ERROR              /* USER-ID NOT IN TABLE
  END-IF
END-BREAK
END-READ
```

Figure 10 – Sample Security Program

In the example above, the user's User-ID is checked in a table to see if the user has access to information about the desired department. If the table indicates that access is permitted, processing continues as usual. If the table does not indicate that access is permitted, the information will not be given to the user.

This implies that there must be a set of programs to maintain the security data on the ADABAS file. The hard-coded approach has some obvious problems:

- Where will the security table be located: VSAM file, copycode, etc.?
- How will the rules be maintained and who will maintain them?
- How will all of the programs requiring this type of logic be maintained?

This leads to two equally unpleasant scenarios:

- Different areas of the organization keeping security rules on separate ADABAS files, causing file proliferation and maintenance problems.

- All areas of the organization keeping their security rules together on one file, causing conflicts to arise over which area should maintain the security rules.

Note: ADABAS Field Level Security is provided as an integral feature in this SECURITRE release.

STRNAT and STRASM eliminate these problems by introducing a simple, yet effective policy: Let the System Security Facility (SSF) keep all the security rules for all definitions, including low-level security rules about screen functions, field value restrictions, and time of day access considerations.

SECURITRE allows these other types of security checks to be handled for NATURAL through program calls to one centralized source, STRNAT. STRASM provides the same access controls as STRNAT, but for non-NATURAL application systems.

STRNAT and STRASM are passed the rule to be validated by the application, and, in turn, pass this rule to the SECURITRE User-Exit-1 to be validated. STRNAT and STRASM relay the result of the request back to the calling application program.

STRNAT and STRASM effectively replace ADABAS Security by Field Value protection, hard-coded checks, and ADABAS-based security rule files or tables.

A program that calls STRNAT/STRASM may set a flag to indicate whether a denied pseudo DSN should be logged as a violation on the system security log. If this flag is not set, violations will be logged.

VI.2 STRNAT

SECURITRE provides an easily maintained, completely effective method of controlling access by field value, among other things. SECURITRE effectively becomes the Server in a security network for NATURAL/ADABAS applications. This Server is reached with the supplied NATURAL subprogram, "STRNAT", which can be invoked with a CALLNAT from any NATURAL application program. This Server is also reached with the assembler module, STRASM, which can be invoked with a standard CALL from most programming languages, including COBOL and PL/1.

STRNAT is passed a "Security Request" by the calling program. The request is in the form of a DSN. This request is validated for the particular user by sending the request to the SSF. STRNAT then passes the result of the request back to the calling program.

VI.2.1 Security by Field Value Example

The following example uses the SECURITRE STRNAT module to control access based on the value of the DEPARTMENT field:

```
      READ PAYROLL-DATA BY DEPARTMENT
      AT BREAK OF DEPARTMENT
        COMPRESS 'PAYROLL.DATA.DEPT' DEPARTMENT INTO
           ENTITY LEAVING NO SPACE
*
*  A SAMPLE RULE WHICH MAY DEVELOP FROM THE ABOVE COMPRESS STATEMENT
*  IS 'PAYROLL.DATA.DEPT1234'.  STRNAT WILL VALIDATE THIS RULE
*  THROUGH THE SYSTEM SECURITY FACILITY.  THERE IS NO NEED TO
*  IDENTIFY THE USER.  SECURITRE WILL DO THIS AUTOMATICALLY.
*
      CALLNAT 'STRNAT' USING
           ENTITY
           ACCESS
           CLASS
           ALLOWED
           COMM-OK
           OTHER
      *
      *
        IF COMM-OK                                  /* IS THE DATABASE UP
            IF ALLOWED                                     /* IS ACCESS ALLOWED
                  IGNORE                                   /* EVERYTHING IS OK
            ELSE
                  PERFORM ACCESS-DENIED                    /* HANDLE ERROR
                  ESCAPE ROUTINE
            END-IF
        ELSE
            PERFORM DATABASE-DOWN                          /* HANDLE ERROR
            ESCAPE ROUTINE
        END-IF
      END-BREAK
      END-READ
```

Figure 11 – STRNAT Security by Field Value Example

The logic above is very easy to maintain. The number of users, the number of departments, and the number of access rules could change repeatedly. No change to the application program would be necessary. The only changes needed would be to the security rules stored on the SSF database.

VI.2.2 Security for NATURAL Example

The following example uses the SECURITRE STRNAT module to control access based on the program being edited:

```
PROGRAM EDIT:

DEFINE DATA
    LOCAL
       01 STRNAT-PARAMETERS
           02 #ENTITY            (A44)          /* DSN TO BE CHECKED BY SAF
           02 #ACCESS            (A1)           /* R = READ, U = UPDATE
           02 #ALLOWED           (L)            /* INDICATES IF ACCESS ALLOWED
           02 #COMM-OK           (L)            /* IS THE DATABASE ACTIVE
           02 #MESSAGE           (A50)          /* ERROR MESSAGE
    LOCAL
           01 #PGM-TYPE          (A1)
           01 #PGM-NAME          (A8)
END DEFINE
*
INPUT #PGM-TYPE #PGM-NAME
*
COMPRESS 'ADABAS.PROD.EDIT.' #PGM-NAME INTO #ENTITY LEAVING NO SPACE
MOVE 'R' TO #ACCESS
*
CALLNAT 'STRNAT' USING STRNAT-PARAMETERS
*
IF #COMM-OK
    IF NOT #ALLOWED                           /* USER NOT PERMITTED TO
       PERFORM ACCESS-DENIED                  /* PERFORM AND EDIT.
       TERMINATE
    END-IF
ELSE
    PERFORM DATABASE-DOWN
    TERMINATE
END-IF
*
STACK TOP COMMAND 'XEDIT' #PGM-TYPE #PGM-NAME
*
END
```

Figure 12 – STRNAT Security For NATURAL Example

VI.2.3 Security for Field Level Security Example

The following example uses the SECURITRE STRNAT module to control access based upon the field(s) being accessed:

```
DEFINE DATA
    LOCAL
01 ENTITY             (A44)          /* DSN TO BE CHECKED
01 ACCESS             (A1)           /* R=READ, U=UPDATE
01 ALLOWED            (L)            /* ACCESS ALLOWED
01 COMM-OK            (L)            /* DATABASE ACTIVE
01 MESSAGE            (A50)          /* ERROR MESSAGE
*

   .
   .
END-DEFINE
*

   .
   .
MOVE 'PAYROLL.DATA.FIELDS.AA.AB.AC' TO ENTITY
MOVE 'R'  TO ACCESS                  /* WANT TO READ IT
*
CALLNAT 'STRNAT' USING
   ENTITY
   ACCESS
   ALLOWED
   COMM-OK
   MESSAGE
*
IF COMM-OK                           /* IS THE DATABASE UP
   IF ALLOWED                        /* IS ACCESS ALLOWED
        IGNORE                       /* EVERYTHING IS OK
   ELSE
        PERFORM ACCESS-DENIED        /* HANDLE ERROR
        ESCAPE ROUTINE
   END-IF
ELSE
   PERFORM DATABASE-DOWN             /* HANDLE ERROR
   ESCAPE ROUTINE
END-IF
   .
   .
END
```

Figure 13 – STRNAT Field Level Security Example

VI.3 Customizing STRNAT

The STRNAT NATURAL subprogram is distributed in source format on the SECURITRE release tape. The subprogram may be customized to allow any of the calling parameters to be changed prior to validation of the security rule. The actual validation of the security rule is accomplished by the NATURAL subprogram STRNATI.

On line 0440 of STRNAT, the variable #STR-DBID should be set to the Database-ID of a database that is actively running SECURITRE. DSNs will be sent to the STRUEX1 of this database for validation, and they will be stored in its DSNPOOL table.

After changes have been made to STRNAT, it must be cataloged and both STRNAT and STRNATI must be placed in a library where they can be found by the calling programs.

The STRNAT calling parameters are described in ***Section VI.1 - STRNAT Calling Parameters*** in the ***SECURITRE Reference Manual***.

VI.4 STRASM

STRASM provides field value access controls that are virtually identical to those provided to NATURAL programs through STRNAT. With STRASM, the calling application generates the rule to be validated, such as "Can the user see records when the value of the Department field is 750?" in a form like the following:

```
PAYROLL.DATA.DEPT.750
```

and passes this rule to STRASM. STRASM sends a request to the SSF to validate the rule. STRASM then passes the result of the request back to the calling program. Consider the following COBOL logic for instance:

```
01  SECURITY-RULE.
    02  SEC-RULE-PART1 PIC X(17).
    02  SEC-RULE-PART2 PIC X(8).
MOVE 'PAYROLL.DATA.DEPT.' TO SEC-RULE-PART1.
MOVE DEPT TO SEC-RULE-PART2.
CALL 'STRASM' USING SECURITY-RULE,ACCESS-TYPE,SSF-CLASS,
     ACCESS-ALLOWED,COMM-OK,MSG
IF ACCESS-ALLOWED ...
```

This is all of the logic necessary to control access to specific values for the DEPT field, regardless of the number of users, departments, or records.

VI.5 Customizing STRASM

The STRASM Assembler subroutine is distributed in source format on the SECURITRE release tape. The subroutine may be customized to allow any of the calling parameters to be changed prior to validation of the security rule. The actual validation of the security rule is accomplished by the Assembler subroutine.

The STRASM calling parameters are described in ***Section VI.2 – STRASM Calling Parameters*** in the ***SECURITRE Reference Manual***.

SECTION VII

USER-EXITS TO SECURITRE

VII.1 Introduction

Note:	This subsection describes the aspects of user-exits to SECURITRE, not user-exits to ADABAS.

SECURITRE allows for interaction of the SECURITRE ADABAS User-Exit-1 with other ADABAS User-Exit-1 modules, as well as providing user-exits to SECURITRE itself.

The general approach SECURITRE takes toward these exits is to allow specification of the load module name of the SECURITRE User-Exit-1 within the "STRPARM" module and then dynamically load other modules from SECURITRE. SECURITRE will then invoke (call) these exit modules when appropriate.

The following subsections detail the specific calling parameters for each SECURITRE user-exit.

VII.2 STREX1 User-Exit

This user-exit may be used to attempt to derive the User-ID based upon site-specific rules. Based upon the TRANID or other CICS related information that would be site specific, it would be possible to identify these commands and to derive a User-ID. The STREX1 exit is activated through the use of the STREX1 parameter of the STRDEF macro. For more information about this parameter, refer to **Section II - Setting Up SECURITRE for ADABAS**.

The STREX1 user-exit to SECURITRE, not an ADABAS User-Exit-1, is invoked in the event that SECURITRE cannot determine the User-ID issuing the command to ADABAS. A typical example would be an asynchronous task under CICS. There is no User-ID associated with these transactions. The calling parameters for STREX1 are:

R1

- +0 The address of the information area. This area will be the USERINFO area for ADABAS. If the USERINFO area is not available the address will be zero.
- +4 The address of the ADABAS Control Block.
- +8 The address of where to place the desired User-ID for this call.
- +12 The address of where to place the indicator (i.e. return code).

R13 The 18 fullword save area.

R14 The return address.

R15 The entry address.

The indicator byte may contain the following values:

00 Indicates that no User-ID has been provided by STREX1. In this case, SECURITRE should handle this command according to the "STRPARM" parms in effect for this ADABAS file. These parameters are NOIDRED and NOIDUPD, which are described in **Section II.4.1 – Setting Up Database and File Security Parameters**.

04 Indicates that a User-ID has been provided by STREX1. In this case, SECURITRE should use this User-ID pointed to by 8(R1), when evaluating this command.

08 Indicates that no User-ID has been provided by STREX1. In this case, SECURITRE should reject this command.

12 Indicates that no User-ID has been provided by STREX1. In this case, SECURITRE should accept this command.

A sample STREX1 can be found in STR.V331.SOURCE(SAMPEX1).

VII.3 STREX2 User-Exit

The STREX2 user-exit to SECURITRE, not an ADABAS User-Exit-2, is invoked after an ADABAS command has passed file level and field level security, if the PROCEX2 parameter for the file has been set to ON. Its purpose is to assign a password to the Additions-3 field of the ADABAS Command Block for use by ADABAS Security By Value processing.

After STREX2 is called, the password is copied from the ADABAS Command Block to the SECURITRE user table, so that multiple calls to STREX2 to retrieve the same information can be avoided. Therefore, STREX2 will be invoked only on the first successful call and not on subsequent calls until the user is removed from the SECURITRE tables.

A sample STREX2 can be found on the SECURITRE release tape. For more information about the member name containing this sample STREX2, refer to **Section IX - Installation**.

The following are calling parameters for STREX2:

R1

+0 The address of the User's ACEE.

+4 The address of the ADABAS Control Block.

R13 The 18 fullword save area.

R14 The return address.

R15 The entry address.

A sample STREX2 can be found in STR.V331.SOURCE(SAMPEX2)

VII.4 STREX3 User-Exit

The STREX3 user-exit to SECURITRE, not to an ADABAS User-Exit-3, is invoked when SECURITRE encounters an unrecoverable situation. When SECURITRE is in an unrecoverable situation, it will issue a WTO indicating the problem and then call STREX3 if provided by the user. Depending on the return indicator from STREX3, SECURITRE may either abend as it did in previous versions, or cause all future commands to be rejected with RC=200 or with the appropriate return codes in SECURITRE for NATURAL.

An example of an unrecoverable situation is when an RTM request to reload parameters is processed, but SECURITRE does not have access to a valid set of parameters. SECURITRE will now issue a WTO with the message "SECURITRE DETECTED INVALID FILE DEFAULTS (STRPARMS)" and then call STREX3 with the abend code 304. Depending on the return indicator from STREX3, it will either abend the database or reject all future incoming calls. If no STREX3 is provided, SECURITRE will cause an abend.

A sample STREX3 can be found on the SECURITRE release tape. For more information about the member name containing this sample STREX3, refer to **Section IX - Installation**.

The following are calling parameters for STREX3:

R1

+0 A half-word containing the abend code.

+4 The address of the 1 byte indicator to be returned.

This indicator advises SECURITRE of the desired action (see below).

R13 The 18 fullword save area.

R14 The return address.

R15 The entry address.

The indicator byte may contain the following values:

00	SECURITRE should reject all subsequent command to this database.
Any non-zero	SECURITRE should abend this database.

A sample STREX3 can be found in STR.V331.SOURCE (SAMPEX3).

This page intentionally left blank.

SECTION VIII

USER-EXIT CO-EXISTENCE

VIII.1 User-Exit-1 Co-Existence

SECURITRE User-Exit-1 can co-exist with any other ADABAS User-Exit-1 module. This "second ADABAS User-Exit-1" will be invoked after SECURITRE has completed processing and is activated through the use of the UEXIT1 parameter of the STRDEF macro. For more information about this parameter, refer to **Section II.10 – Interfaces to Other Products**.

The UEXIT1 module should be unaffected by the presence of SECURITRE. All registers on entry to the UEXIT1 program will be the same as the values they were on entry to SECURITRE, with the exception of Registers 13, 14, and 15. These registers will be set as follows:

R13 The address of an 18 fullword save area within SECURITRE. The UEXIT1 program should follow standard IBM register conventions, save the register on entry, and restore the registers on exit. This register would normally point at a save area within ADABAS.

R14 The address of the return point to SECURITRE. This register would normally contain a return point to ADABAS.

R15 The address of the entry point to the UEX1 program. This register would normally be the same as if ADABAS had called this user-exit directly.

This exit may reject a command that SECURITRE has allowed, but if SECURITRE has rejected the command, the command will be rejected regardless of the action of UEXIT1. For further information regarding the functions of an ADABAS User-Exit-1, consult the appropriate ADABAS reference manual.

VIII.2 User-Exit-4 Co-Existence

TSI provides the ability for multiple ADABAS User-Exit-4 routines to co-exist transparent to each other. This is accomplished through an ADABAS User-Exit-4 (TSIUEX4) that acts as a "dispatcher" module. This "dispatcher" will dynamically load all required ADABAS User-Exit-4 routines during initialization processing and then invoke each of the ADABAS User-Exit-4 programs in turn.

Each user-exit will receive control with the same registers in effect that ADABAS would have provided, with the following exceptions:

R13 The address of an 18 fullword save area within the "dispatcher" module. This register would normally point at a save area within ADABAS.

R14 The address of the return address to the "dispatcher" module. This register would normally contain a return point to ADABAS.

Effects On Logging

The primary purpose of an ADABAS User-Exit-4 routine is to control the logging of data to the ADABAS Command Log. This is accomplished by passing a "flag" back to ADABAS to indicate if the command should be logged or not. This is accomplished by an indicator byte in the parameter list.

TSIUEX4 will return the flag from the last ADABAS User-Exit-4 invoked by the dispatcher to ADABAS.

For example, assume that TSIUEX4 has been instructed to "dispatch" three user-exits, and these user-exits have indicated a logging action as follows:

STRUEX4	SECURITRE	(NOLOG)
USRUEX4	user program	(NOLOG)
TRMUEX4R	TRIM	(LOG)

The user-exit routines are "dispatched" by TSIUEX4 in the order specified. Therefore, ADABAS will log the command as a result of the example above.

VIII.3 TSIEX4P Statement

The purpose of the TSIEX4P statement is to specify the names of the ADABAS User-Exit-4 programs to be loaded by TSIUEX4 and the order of their execution.

The format for the TSIEX4P parameter is standard macro assembler format:

- Opcode in column 10
- One or more spaces
- Operands up to column 71, separated by commas
- Continuation symbol ("x") in column 72
- Continuation lines start in column 16

TSIUEX4 executes the various User-Exit-4s specified in the TSIEX4P statement in a predetermined order, as listed below:

First:	UEX4SC	SECURITRE User-Exit-4 (necessary if TRIM is not in use)
Next:	UEX4N1	First non-TSI User-Exit-4 (if any)
	UEX4N2	Second non-TSI User-Exit-4 (if any)
	UEX4N3	Third non-TSI User-Exit-4 (if any)
Last:	UEX4TR	TRIM User-Exit-4 (if TRIM is in use)

SECURITRE User-Exit-4 (UEX4SC) is always executed first, if used, and TRIM User-Exit-4 (UEX4TR) is always executed last (if TRIM is installed).

VIII.4 TSIEX4P Parameters

UEX4SC The name of the SECURITRE User-Exit-4 routine to be invoked by the "dispatcher." The SECURITRE User-Exit-4 will be invoked before all other User-Exit-4s specified in the TSIEX4P statement.

Valid Values: a valid load module name
Default Value: no default value
Assigned By: TSIEX4P only

UEX4N1 The name of the first Non-TSI User-Exit-4 routine to be invoked by the "dispatcher." This User-Exit-4 will be invoked immediately after the SECURITRE User-Exit-4 specified in the UEX4SC parameter has completed processing.

Valid Values: a valid load module name
Default Value: no default value
Assigned By: TSIEX4P only

UEX4N2 The name of the second Non-TSI User-Exit-4 routine to be invoked by the "dispatcher." This User-Exit-4 will be invoked after the ADABAS User-Exit-4 specified in the UEX4N1 parameter has completed processing.

Valid Values: a valid load module name
Default Value: no default value
Assigned By: TSIEX4P only

UEX4N3 The name of the third Non-TSI User-Exit-4 routine to be invoked by the "dispatcher." This User-Exit-4 will be invoked after the ADABAS User-Exit-4 specified in the UEX4N2 parameter has completed processing.

Valid Values: a valid load module name
Default Value: no default value
Assigned By: TSIEX4P only

UEX4TR The name of the TRIM User-Exit-4 routine to be invoked by the "dispatcher." The TRIM User-Exit-4 routine will be invoked after all other User-Exit-4s have completed processing.

Valid Values: a valid load module name
Default Value: no default value
Assigned By: TSIEX4P only

(continued on next page)

(continued from previous page)

PRINT Indicates to SECURITRE whether or not the user wants TSIEX4P assembled with macro expansions generated. Macro notes (MNOTES) generated by TSIEX4P macros will be printed in the listing when either "GEN or "NOGEN" is specified.

GEN causes macro expansions to be printed in the listing and will take up a significant amount of paper.

NOGEN does not cause macro expansions to be printed and conserves paper.

Valid Values: GEN, NOGEN
Default Value: NOGEN
Assigned By: TSIEX4P only

VIII.5 TSIEX4P Statement Sample

The following sample parameters will set the User-Exit-4 "dispatcher" to function in an ADABAS environment:

Example 1:

```
                                                                    Col. 72
TSIEX4P  UEX4SC=STRUEX4,UEX4N1=MYPGM1,UEX4N2=MYPGM2,                  x
         UEX4N3=MYPGM3,UEX4TR=TRMUEX4R,                               x
         PRINT=GEN
```

- **UEX4SC=STRUEX4**
  The "dispatcher" will load and initiate SECURITRE ADABAS User-Exit-4 STRUEX4 before invoking any other User-Exit-4s specified in the TSIEX4P statement.

- **UEX4N1=MYPGM1**
  The "dispatcher" will load and initiate a valid ADABAS User-Exit-4 called MYPGM1 after STRUEX4 has completed processing.

- **UEX4N2=MYPGM2**
  The "dispatcher" will load and initiate a valid ADABAS User-Exit-4 called MYPGM2 after MYPGM1 has completed processing.

- **UEX4N3=MYPGM3**
  The "dispatcher" will load and initiate a valid ADABAS User-Exit-4 called MYPGM3 after MYPGM2 has completed processing.

- **UEX4TR=TRMUEX4R**
  The "dispatcher" will load and initiate TRIM ADABAS User-Exit-4 called TRMUEX4R after MYPGM3 has completed processing.

- **PRINT=GEN**
  SECURITRE will inform the assembler to print macro generated code in the assembler listing, generating a significant number of printed pages.

Example 2:

```
*
*        ALLOW SECURITRE,TRIM, AND MY USER-EXIT-4 TO CO-EXIST
*
         TSIEX4P UEX4SC=STREXIT4,           SECURITRE USER-EXIT-4      X
                 UEX4N1=USEREX4,            OTHER USER-EXIT-4          X
                 UEX4TR=TRMUEX4R,           TRIM USER-EXIT-4           X
                 PRINT=NOGEN                CONSERVE PAPER
*
         END
```

When the parameters in Example 2 are assembled, the assembler generates the listing below. The listing details what parameters have been specified to TSIEX4P.

```
*
*        ALLOW SECURITRE, TRIM, AND MY USER-EXIT-4 TO CO-EXIST

         TSIEX4P UEX4SC=STREXIT4,             SECURITRE USER-EXIT-4      X
                 UEX4N1=USEREX4,              OTHER USER-EXIT-4          X
                 UEX4TR=TRMUEX4R,             TRIM USER-EXIT-4           X
                 PRINT=NOGEN                  CONSERVE PAPER
*
*
             PRINT NOGEN
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*             T S I E X 4 P      -  V3.3.1
*
*      PRODUCT OF TREEHOUSE SOFTWARE, INC.
*              ALL RIGHTS RESERVED.
*
** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        UEX4N1 = USEREX4
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        UEX4N2 NOT ACTIVE
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        UEX4N3 NOT ACTIVE
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        UEX4TR = TRMUEX4R
*        T R I M  USER-EXIT-4 ACTIVE
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        UEX4SC = STREXIT4
*        S E C U R I T R E  USER-EXIT-4 ACTIVE
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
          END
```

VIII.6 Relationship of TSIUEX4 to Other User-Exit-4s

The diagram below depicts the relationship between the "dispatcher" driver, TSIEX4DR, the parameters to the dispatcher (TSIEX4PR), the link-edit of these two modules as TSIUEX4, and the various subordinate ADABAS User-Exit-4 routines (STRUEX4, USERX4I, USERXYZ, USERX43, and TRMUEX4R).

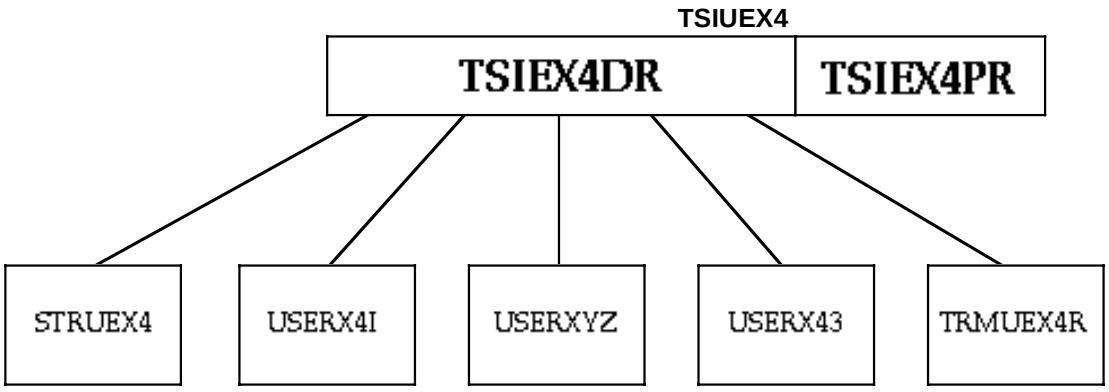

Figure 14– Relationship of TSIUEX4 to Other User-Exit-4s

VIII.7 Activating User-Exit-4 Dispatcher

In order to activate the TSIUEX4 User-Exit-4, follow the procedures in **Section IX.4.10 - Activate SECURITRE for ADABAS**. However, the ADARUN parameter related to UEX4 must be stated as:

- `ADARUN UEX4=TSIUEX4`

This page intentionally left blank.

SECTION IX

INSTALLATION

Note:	If SECURITRE 3.3.1 is already installed, skip to section IX.4.2 for information on installing zaps and fixes.

IX.1 Introduction

SECURITRE may be installed and executed on OS/390, MVS, MVS/XA, MVS/ESA, or z/OS on an IBM 370 or an IBM compatible mainframe. Installation requires only a few cylinders of disk space.

SECURITRE requires no Zaps to any operating system, teleprocessing system, or to ADABAS, NATURAL, or their associated software.

SECURITRE will issue all allocate requests above the 16M line, if ADABAS is running above the line (i.e., ADARUN has been re-linked AMODE=31).

Using ADABAS User-Exit-1 instead of ADALINK, as the primary security control point for ADABAS file security ensures that SECURITRE cannot be bypassed. SECURITRE uses an ADABAS User-Exit-B to obtain a User-ID from ACF2, RACF, or TOP-SECRET (TSS), but all security checks are made in the SECURITRE ADABAS User-Exit-1.

SECURITRE provides macros that generate parameters referred to as STRPARMs, including the SAF entity (DSN) to be used when checking each file. This means that SECURITRE will conform to the site's rules. These site dependencies are accommodated using the SECURITRE macro generated parameters STRDEF and STRFNR. For more information, refer to **Section II - Setting Up SECURITRE for ADABAS**.

Control over the use of NATURAL is provided by including certain SECURITRE-supplied modules when the NATURAL module is linked. The STNPARM parameters that are used to specify which SECURITRE for NATURAL features will be implemented are included at that time. For more information, refer to **Section III - Setting Up SECURITRE for NATURAL**.

The first step in a SECURITRE installation involves preparing the System Security Facility (SSF) to accept authorization calls from SECURITRE. The exact procedures performed during this step depend upon the SSF in use. That is, a different series of actions is performed for RACF than for ACF2 or TOP SECRET. This is documented in the Integration of SECURITRE with the SSF section of this manual. The section provides relevant headings related to each specific security package.

Once the SSF is prepared to accept STR authorization calls, the next step involving the installation of the components of SECURITRE. SECURITRE has five separate components: ADABAS File Security, the SECURITRE Real-Time Monitor (RTM), ADABAS Utility Security, and SECURITRE for NATURAL.

Not every customer will use all of the components**. However, ADABAS File Security is the only mandatory component, and it must be installed first.**

This component allows the SSF to control access to ADABAS Files using site specific requirements by linking individual SECURITRE components to create STRUEX1 and STRUEX4, setting the ADARUN parameters UEX1-STRUEX1 and UEX4-STRUEX4 to cause ADABAS to call these SECURITRE user-exits for every ADABAS call, and linking SECURITRE's User-Exit-Bs with the adalink for each teleprocessing environment used (e.g., Batch, CICS, COM-PLETE, and TSO). ADABAS File Security also defines some SECURITRE ADABAS File parameters and defines some rules to the SSF based on the SECURITRE ADABAS File parameters.

If SECURITRE for NATURAL is used, it controls six levels of security within NATURAL: NATURAL Session Initialization, LOGON, DDM, PROGRAM, RUN Security, and NATURAL Utility Security. It is installed by linking several SECURITRE modules with the NATURAL nucleus, making minor changes to the NATPARMS, defining some SECURITRE for NATURAL parameters, and defining some rules to the SSF based on the SECURITRE for NATURAL Parameters.

If the SECURITRE Real-Time Monitor is used, the SECURITRE RTM NATURAL modules are required to be installed on only one database, preferably the TEST database. However, they may be installed on multiple databases. Operation of the RTM is described in *Section V - Real-Time Monitor of the SECURITRE Reference Manual*. The SECURITRE RTM NATURAL Modules are distributed for execution using NATURAL 3.1.6 and above.

If SECURITRE ADABAS Utility Security is used ADABAS Utilities are controlled using a SECURITRE supplied front-end to ADARUN. For more information, refer to **Section IV - SECURITRE for ADABAS Utilities**.

Other data controls from NATURAL applications can be performed using the STRNAT facility, and other data controls from non-NATURAL applications can be performed using the STRASM facility. For more information, refer to **Section VI - Internal Application Security Features (STRNAT and STRASM)**.

The following checklist for the installation of SECURITRE's ADABAS File Security under ADABAS is provided for the installer's convenience. For more specific instructions, refer to the installation manual.

Following is a list of steps to install the **required** ADABAS File Security. Full installation instructions are in the Installation of SECURITRE for ADABAS File Security Section of this manual.

STEP	FUNCTION	SECTION	REQUIRED
A1	Prepare SSF for SECURITRE	IX.2	Y
A2	Load Datasets	IX.4.1	Y
A3	Install zaps and fixes	IX.4.2	Y
A4	Assemble STRIOR	IX.4.3	Y
A5	Apply the authorization Zap and any fix Zaps	IX.4.4	Y
A6	Link-edit STRUEX1	IX.4.5	Y
A7	Code STRPARM	IX.4.6	Y
A8	Assemble/Link STRPARM	IX.4.7	Y
A9	APF-Authorize	IX.4.8	Y
A10	Install TSIRDC	IX.4.9	CICS ONLY
A11	Install ADALINK EXITS (UEXITB)	IX.4.10	Y
A12	Activate SECURITRE for ADABAS	IX.4.11	Y
A13	Install Co-existing User-Exits	IX.4.12	N

Repeat steps A7 through A13 for other databases.

Following is a list of steps to install the **optional** ADABAS Utility Security. Full installation instructions are in **Section X.5 - Install ADABAS Utility Security**.

STEP	FUNCTION	SECTION	REQUIRED
U1	Apply the authorization Zap and any fix Zaps	IX.5.1	Y
U2	Link-edit STURUNL	IX.5.2	Y
U3	Link-edit ADARUN front-end	IX.5.3	Y
U4	Modify STRPARM	IX.5.4	Y
U5	Assemble/Link STRPARM	IX.5.5	Y
U6	APF-Authorize	IX.5.6	Y

Repeat steps U3 through U6 for other ADARUNs.

Following is a list of steps to install the **optional** SECURITRE for NATURAL. Full installation instructions are in **Section X.6 - Install SECURITRE for NATURAL**.

STEP	FUNCTION	SECTION	REQUIRED
N1	Apply any fix Zaps	IX.6.1	Y
N2	Code STNPNAT	IX.6.2	Y
N3	Assemble/Link STNPNAT	IX.6.3	Y
N4	Modify the NATPARM module	IX.6.4	Y
N5	Assemble/Link NATPARM module	IX.6.5	Y
N6	Link-edit a NATURAL nucleus	IX.6.6	Y
N7	Activate SECURITRE for NATURAL	IX.6.7	Y
N8	Install SECURITRE for NATURAL Logon Security	IX.6.8	N

Repeat steps N2 through N8 for each NATURAL nucleus that will be secured by SECURITRE for NATURAL.

Following is a list of steps to install the **optional** SECURITRE Real-Time Monitor (RTM). Full installation instructions are in the Install SECURITRE for NATURAL section of this manual.

Note: It is recommended that the RTM be installed. Without the RTM, the database where SECURITRE is installed will need to be brought down and up to allow SECURITRE to access the new parms if the "STRPARMS" or the SSF rules are modified.

STEP	FUNCTION	SECTION	REQUIRED
R1	NATLOAD the RTM modules	IX.7.1	Y
R2	Execute STRUXCPY	IX.7.2	Y
R3	Verify the installation	IX.7.3	Y
R4	Modify the "STRPARM" module	IX.7.4	Y
R5	Assemble/link the new "STRPARM" module	IX.7.5	Y
R6	Modify the NATPARM module	IX.7.6	Y
R7	Assemble/Link the NATPARM module	IX.7.7	Y
R8	Link-edit a NATURAL nucleus	IX.7.8	Y
R9	Define rules to SSF	IX.7.9	Y

IX.2 Integration of SECURITRE with the SSF

Integration of SECURITRE with the SSF is a simple process for the Security Administrator. This process consists of defining ADABAS and NATURAL as resources to the SSF, permitting SECURITRE to pass requests through to the SSF. This process is different for each of the three major SSFs.

SECURITRE will send requests to the SSF for the following types of access:

FILE SECURITY
SECURITRE for ADABAS Nucleus SECURITRE requests UPDATE (X'04') access for A1, A4, E1, E4, N1, and N2 commands.

READ access (X'02') is requested for all other commands that require an ADABAS file number.

UTILITY SECURITY
All calls made to the SSF for Utility Security request UPDATE access.

SECURITRE FOR NATURAL
All calls made to the SSF by SECURITRE for NATURAL request READ access, with the exception of the Privileged LOGON.

RTM SECURITY
All calls made to the SSF for RTM Security request READ access.

IX.2.1 ACF2 Installation

Integrating SECURITRE with ACF2 involves selecting a few ACF2 options and defining a SAF Protection List entry. This is a three-step process.

Step One

The first step creates the SAF Protection List entry and opens the "SAF gateway" to ACF2 for SECURITRE by issuing commands from TSO.

The Security Administrator invokes ACF2 by entering:

```
ACF
```

Then, the General System Options are selected by entering:

```
SET CONTROL(GSO)
```

Then, in order to open the gateway that SECURITRE will use to communicate with ACF2, the user must enable the SAF protocol by entering the following:

```
CH OPTS SAF
```

The SAF Protection List entry for ADABAS with the qualifier "ADARUN" must be defined. (The entry should define the ADARUN subsystem with a control point of ADARUN and specify that all resource classes are acceptable.)

```
IN SAFPROT.ADARUN SUBSYS(ADARUN) CNTLPTS(ADARUN) + CLASSES(-)
```

If the ADARUN module has been renamed, substitute the new name for "ADARUN" in the command above. SAFSAFE reduces the overhead resulting from enabling SAF in ACF2 by preventing SAF calls made by those subsystems and control points not specified in the SAFPROT list. The Security Administrator must ensure that the ADARUN subsystem and control point are properly listed in the SAFPROT list or SECURITRE will not be able to issue calls to ACF2. A more detailed discussion of these topics can be found in the ACF2 System Programmer Guide section of this manual.

The ACF2 session is then ended by entering:

```
E
```

Step Two

The second step in preparing ACF2 to use SECURITRE is to make certain that the Logon-ID of the ADABAS region has MUSASS privileges. These privileges allow ADABAS to operate as a Multi-User Single Address Space System. For more information, refer to the ACF2 System Programmer Guide section of this manual.

Step Three

The third step should be implemented once the steps above are completed. The Security Administrator must then make ACF2 recognize the changes that have been made. This is accomplished through the following commands:

```
F ACF2,REFRESH(OPTS)
F ACF2,REFRESH(SAFPROT)
```

The Security Administrator should then reply to the JES prompts with a valid Logon-ID containing the REFRESH privilege and the password of that Logon-ID.

Note: For versions of ACF2 prior to Version 5.1, the MUSUPDT keyword does not exist. The SECURITY and ACCOUNT keywords must be used in place of MUSUPDT.

IX.2.2 TOP SECRET Installation

To integrate SECURITRE with TOP SECRET, the Security Administrator must define the facility "ADABAS" to TOP SECRET.

The following is an example of a facility that defines the ADABAS Master Facility to TOP SECRET. Sites should use their defaults when defining the facility.

```
INITPGM=ADA  ID=U TYPE=25
ATTRIBUTES=IN-
USE,ACTIVE,SHRPRF,NOASUBM,NOTENV,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=NOLUMSG,NOSTMSG,SIGN(M),NOPSEUDO,INSTDATA,NORNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,NOMRO,WARNPW,NOTSOC
ATTRIBUTES=NOTRACE,NOLAB,NODORMPW,NONPWR,NOIMSXTND
MODE=WARN
LOGGING=SMF,MSG,SEC9
UIDACID=8 LOCKTIME=000 DEFACID=*NONE* KEY=8
```

In the example above, the master facility is being brought up in WARN mode to facilitate the installation. This should be changed to FAIL mode as soon as possible.

If ADABAS is executed as a started task, the Security Administrator should add the Master Facility to the STC ACID:

```
TSS ADD USERID(STC) MASTFAC(ADABAS)
```

Integration with TOP SECRET is complete once the Security Administrator has defined users of the new ADABAS Master Facility to TOP SECRET:

```
TSS ADD USERID(USERS) FACILITY (ADABAS)
```

IX.2.3 RACF Installation

To enable SECURITRE to operate under ACF2 and TOP SECRET, special setup procedures are involved as previously indicated. However, no special setup steps or changes need to be made to RACF to allow SECURITRE to function properly.

IX.3 OS (OS/390, MVS, MVS/XA, MVS/ESA, z/OS) Installation

This distribution of SECURITRE contains the following datasets.

DSN #	Dataset NAME	Library Dataset	Member Contents	Generated Format	Storage Requirement
1	STR.V331 SOURCE	Source Library	several JCL members, source code, macros, and sample parameters*	IEBCOPY format	two cylinders 3380 disk space, or equivalent
2	STR.V331 LOAD	Load Library	load members that will be link-edited into the installation load library*	IEBCOPY Format	four cylinders 3380 disk space, or equivalent
3	STR.V331 RTMNTLD	NATLOAD Dataset	SECURITRE RTM NATURAL programs	NATURAL NATUNLD format	three cylinders 3380 disk space, or equivalent
4	STR.V331. README	Readme Dataset	Information on all available official zaps and fixes	IEBGener format	Ten tracks 3380 disk space or equivalent
5	STR.V331. FIX.INSTAL L	Fix installation instructions	Information on installing available zaps and fixes	IEBGener	Ten Tracks 3380 disk space or equivalent

* Contents listed on the following pages.

Note: The JCL included on the installation tape and in this reference manual is not intended to be run as-is. Some modification will be required to make the procedures conform to site specifications.

The Source Library is generated in IEBCOPY format and contains the following members:

JCL Members

```
JCLASM         (JCL to do required assemblies)
JCLLKDA        (JCL to link-edit ADARUN)
JCLLKDU1       (JCL to link-edit Utility Security - step 1)
JCLLKDU2       (JCL to link-edit Utility Security - step 2)
JCLLKED1       (JCL to link-edit SECURITRE User-Exit-1
JCLLOAD        (JCL to load SECURITRE datasets to disk)
JCLNLOAD       (JCL to load RTM with LOAD)
```

Source Code, Macros

```
SAMPEX1        (Sample STREX1 User-Exit to SECURITRE-refer to Section IX)
SAMPEX2        (Sample STREX2 User-Exit to SECURITRE)
SAMPEX3        (Sample STREX3 User-Exit to SECURITRE)
TSISTRX1       (Sample  STREX1  User-Exit  to  SECURITRE-refer  to  module  for
               details)
STNDDM         (Macro for generating SECURITRE for NATURAL parms)
STNFILE        (Macro for generating SECURITRE for NATURAL parms)
STNFINI        (Macro for generating SECURITRE for NATURAL parms)
STNLIB         (Macro for generating SECURITRE for NATURAL parms)
STNMIX         (Macro for generating SECURITRE for NATURAL parms)
STNPARM        (Macro for generating SECURITRE for NATURAL parms)
STNTEST        (Macro for generating SECURITRE for NATURAL parms)
STRASM         (Code to send request to SSF for authorization)
STRDEF         (Macro for generating SECURITRE default settings)
STRFNR         (Macro for generating SECURITRE settings by file)
STRIOR         (Operating System dependent code)
STRUEXBC       (User-Exit-B for COM-PLETE)
STRUNQ         (Macro for generating SECURITRE for NATURAL parms)
TSIEX4P        (Macro for generating dispatcher parms -  refer to Section IX)
```

Sample Parameters

```
STNPNAT        (Sample SECURITRE for NATURAL parameters)
STP00007       (Sample SECURITRE parameters)
TSIEX4PR       (Sample TSIUEX4 dispatcher parms - refer to Section IX)
```

The Load Library is generated in IEBCOPY format and contains the following members:

```
STRABEND         (User-Exit-1 Component)
STRDATA          (User-Exit-1 Component)
STRDSNB          (User-Exit-1 Component)
STRERR           (User-Exit-1 Component)
STRFFNR          (User-Exit-1 Component, and
                  Utility Security Component)
STRFLSP          (User-Exit-1 Component)
STRINIT          (User-Exit-1 Component)
STRINTV          (User-Exit-1 Component)
STRMAIN          (User-Exit-1 Component)
STRMON           (User-Exit-1 Component)
STRPUPD          (User-Exit-1 Component)
STRSUB1          (User-Exit-1 Component)
STRSUB2          (User-Exit-1 Component)
STRSUB3          (User-Exit-1 Component)
STRTAB           (User-Exit-1 Component)
STRTAB2          (User-Exit-1 Component)
STRTRAC          (User-Exit-1 Component)
STRTRCI          (User-Exit-1 Component)
STRZAP           (User-Exit-1 Component)

STUDATA          (Utility Security Component)
STUFAIL          (Utility Security Component)
STUFFNR          (Utility Security Component)
STUINIT          (Utility Security Component)
STUOPER          (Utility Security Component)
STURUN           (Utility Security Component)
STUSTR           (Utility Security Component)
STUTAB5          (Utility Security Component - ADABAS 5)
STUTAB6          (Utility Security Component - ADABAS 6)
STUTAB7          (Utility Security Component – ADABAS 7)
STUZAP           (Utility Security Component)

STRUEXBB         (User-Exit-B for Batch/TSO)
STRUEXBI         (User-Exit-B for IMS)

STRUEXB3         (User-Exit-B for CICS)
STRUEXB5         (User-Exit-B for COM-PLETE 4.5+)

STRUEX4          (User-Exit-4)
TSIEX4DR         (User-Exit-4 Dispatcher)

TSIRDC31         (RDC Exit for NATURAL 3.1.6 component)
TSIRDC41         (RDC Exit for NATURAL 4.1 component)
TSIRDC42         (RDC Exit for NATURAL 4.2 component)

STNA             (SECURITRE for NATURAL 3.1.6 component)
STNB             (SECURITRE for NATURAL 3.1.6 component)
STND             (SECURITRE for NATURAL 3.1.6 component)
STNPRM           (SECURITRE for NATURAL 3.1.6 component)
STNZAPL          (SECURITRE for NATURAL 3.1.6 component)
STNZAPP          (SECURITRE for NATURAL 3.1.6 component)
STNZAPS          (SECURITRE for NATURAL 3.1.6 component)
STRACHEK         (SECURITRE for NATURAL 3.1.6 component)
STRLGN           (SECURITRE for NATURAL 3.1.6 component)

STN4A            (SECURITRE for NATURAL 4.1 component)
STN4B            (SECURITRE for NATURAL 4.1 component)
STN4D            (SECURITRE for NATURAL 4.1 component)
STN4E            (SECURITRE for NATURAL 4.1 component)
STN4FREE         (SECURITRE for NATURAL 4.1 component)
STN4GET          (SECURITRE for NATURAL 4.1 component)
STN4PRM          (SECURITRE for NATURAL 4.1 component)
STN4ZAPL         (SECURITRE for NATURAL 4.1 component)
STN4ZAPP         (SECURITRE for NATURAL 4.1 component)
STN4ZAPS         (SECURITRE for NATURAL 4.1 component)
STN4RCHK         (SECURITRE for NATURAL 4.1 component)
STN4LGN          (SECURITRE for NATURAL 4.1 component)

STN42A           (SECURITRE for NATURAL 4.2 component)
```

```
STN42B          (SECURITRE for NATURAL 4.2 component)
STN42D          (SECURITRE for NATURAL 4.2 component)
STN42E          (SECURITRE for NATURAL 4.2 component)
STN42FRE        (SECURITRE for NATURAL 4.2 component)
STN42GET        (SECURITRE for NATURAL 4.2 component)
STN42PRM        (SECURITRE for NATURAL 4.2 component)
STN42ZAL        (SECURITRE for NATURAL 4.2 component)
STN42ZAP        (SECURITRE for NATURAL 4.2 component)
STN42ZAS        (SECURITRE for NATURAL 4.2 component)
STN42RCH        (SECURITRE for NATURAL 4.2 component)
STN42LGN        (SECURITRE for NATURAL 4.2 component)

STRASMI         (Used with STRASM to check DSN authorization)
```

IX.4 Installation of SECURITRE for ADABAS File Security

The following subsections describe how SECURITRE must be integrated with ADABAS to use ADABAS File Security. In general, this process consists of physically installing SECURITRE on the computer system, tailoring SECURITRE to site specifications, link-editing individual load modules, applying any authorization and fix Zaps, link-editing appropriate modules to the ADABAS load libraries, and activating SECURITRE.

The process varies depending upon the operating system in use at the installation. The installation instructions for the appropriate operating system must be used to ensure proper installation and functioning of SECURITRE.

SECURITRE should first be installed on an "experimental" or Test database. The installation procedure steps for ADABAS File SECURITY, which are described in detail later, for OS (OS/390, MVS, MVS/XA, MVS/ESA, z/OS) are as follows:

A1 Prepare the SSF to accept SECURITRE commands. (Refer to section IX.2.)

A2 LOAD the SECURITRE datasets to the disk libraries. (Refer to section IX.4.1.)

A3 Install all zaps/fixes (Refer to section IX.4.2)

A4 Assemble the STRIOR module. (Refer to section IX.4.3.)

A5 Apply the authorization Zap and any fix Zaps to individual load modules. (Refer to section IX.4.4.)

A6 Link-edit individual load modules to create STRUEX1. (Refer to section IX.4.5.)

A7 Code an "STRPARM" parameter module. (Refer to section IX.4.6 for installation instructions.) Before coding an "STRPARM" module refer to ***Section II - SECURITRE for ADABAS NUCLEUS*** in the ***SECURITRE Reference Manual*** for the possible values for this parameter module. For some examples of uses of this parameter module, refer to **Section II - Setting Up SECURITRE for ADABAS**.

A8 Assemble and link-edit the "STRPARM" parameter module into the load library. (Refer to section IX.4.7.)

A9 APF-authorize ADABAS and SECURITRE ADABAS File Security modules. (Refer to section IX.4.8.)

A10 Install TSIRDC (Refer to section IX.4.9.)

A11 Install ADABAS Link Routine exits. (Refer to section IX.4.10.)

A12 Activate SECURITRE for ADABAS. (Refer to section IX.4.11.)

A13 Once operational, the issue of "co-existing User-Exit-1s, User-Exit-4s and User-Exit-Bs" can be dealt with. For more information, refer to **Section IX.4.12 - Co-existing User-Exits**. (Refer to section IX.4.12.)

Repeat steps A7 through A13 for all other databases to be secured.

Use this checklist when installing ADABAS File Security:

✓	**STEP**	**FUNCTION**	**SECTION**	**REQUIRED**
	A1	Prepare SSF for SECURITRE	IX.2	Y
	A2	Load Datasets	IX.4.1	Y
	A3	Install Zaps/Fixes	IX.4.2	
	A4	Assemble STRIOR	IX.4.3	Y
	A5	Apply the authorization Zap and any fix Zaps	IX.4.4	Y
	A6	Link-edit STRUEX1	IX.4.5	Y
	A7	Code STRPARM	IX.4.6	Y
	A8	Assemble/Link STRPARM	IX.4.7	Y
	A9	APF-Authorize	IX.4.8	Y
	A10	Install TSIRDC	IX.4.9	CICS ONLY
	A11	Install ADALINK EXITS (UEXITB)	IX.4.10	Y
	A12	Activate SECURITRE for ADABAS	IX.4.11	Y
	A13	Install Co-existing User-Exits	IX.4.12	N

Repeat steps A7 through A13 for other databases.

IX.4.1 Load Datasets

SECURITRE is distributed on a 3490 cartridge with standard labels or via the internet. If installing from an internet distribution, refer to section IX.4.1.1 to load the datasets. If installing from a cartridge, refer to section IX.4.1.2 to load the datasets.

IX.4.1.1 Load datasets from a web or email distribution

You must have an FTP server running on your mainframe and an FTP client running on your PC in order to transfer these files.

SECURITRE is distributed as a ZIP file containing the following files:

File	Description
STRv331RN.PDF	SECURITRE Release Notes
STRv331Ref.PDF	SECURITRE Reference Manual
STRv331Admin.PDF	SECURITRE Administration Manual
STRV331.RTMNTLD.XMT	NATURAL source/object code in NATLOAD format
STRV331.SOURCE.XMT	PDS containing source code
STRV331.LOAD.XMT	PDS containing load modules
STRV331.README.TXT	Information on zaps and fixes created after the initial release.
STRV331.FIX.INSTALL.TXT	Information on installing available official zaps and fixes

Installation Procedure:

Summary of installation procedure:

1) Allocate datasets
2) Load datasets
3) SECURITRE installation

Allocate Datasets

Allocate the following datasets

Dataset	DCB Information
TEMP.STR.V331.SOURCE	RECFM=FB,LRECL=80,BLKSIZE=3120, SPACE=(CYL,(2,2)),DSORG=PS
TEMP.STR.V331. LOAD	RECFM=FB,LRECL=80,BLKSIZE=3120, SPACE=(CYL,(4,4)),DSORG=PS
TEMP.STR.V331.RTMNTLD	RECFM=FB,LRECL=80,BLKSIZE=3120, SPACE=(CYL,(3,3)),DSORG=PS
STR.V331.README	RECFM=FB,LRECL=80,BLKSIZE=8000, SPACE=(CYL,(1,1)),DSORG=PS
STR.V331.FIX.INSTALL	RECFM=FB,LRECL=80,BLKSIZE=8000, SPACE=(CYL,(1,1)),DSORG=PS

Load Datasets
In BINARY mode transfer files:

STRV331.SOURCE.XMT to TEMP.STR.V331.SOURCE
STRV331.LOAD.XMT to TEMP.STR.V331.LOAD
STRV331.RTMNTLD.XMT to TEMP.STR.V331.RTMNTLD

Once the binary transfers are complete, issue the following commands from the TSO command line:

Command	Restore Parameters
receive indataset('temp.str.v331.source')	Dsname('str.v331.source')
receive indataset('temp.str.v331.load')	Dsname('str.v331.load')
receive indataset('temp.str.v331.rtmntld')	Dsname('str.v331.rtmntld')

Once all datasets have been transmitted and successfully received, the TEMP.* datasets may be deleted.

In ASCII Mode
Transfer STRV331.README.TXT to STRV331.STR.V331.README and STRV331.FIX.INSTALL.TXT to STRV331.STR.V331.FIX.INSTALL.

SECURITRE Installation
Skip to section IX.4.2.

IX.4.1.2 Load datasets from a cartridge distribution

SECURITRE cartridge map:

Label	Dataset name	Block Size	LRECL	RECFM	Format
1	STR.V331.SOURCE	6160	80	FB	IEBCOPY
2	STR.V331.LOAD	6447	0	U	IEBCOPY
3	STR.V331.RTMNTLD	2564	256	VB	NATLOAD
4	STR.V331.README	8000	80	FB	IEBGENER
5	STR.V331.FIX.INSTALL	8000	80	FB	IEBGENER

The following sample JCL can be used to allocate space and load all datasets contained on the release tape:

```
//*            MEMBER(JCLALLOC)
//*            ALLOCATE DISK SPACE
//*
//ALLOC1       EXEC    PGM=IEFBR14
//SECTRE1      DD      DSN=STR.V331.SOURCE,SPACE=(CYL,(2,0,5)),
//                     UNIT=SYSDA,VOL=SER=XXXXXX,
//                     DISP=(,CATLG,DELETE),
//                     DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PO)
//SECTRE2      DD      DSN=STR.V331.LOAD,SPACE=(CYL,(5,0,10)),
//                     UNIT=SYSDA,VOL=SER=XXXXXX,
//                     DISP=(,CATLG,DELETE),
//                     DCB=(RECFM=U,BLKSIZE=6447,DSORG=PO)
//SECTRE3      DD      DSN=STR.V310.README,SPACE=(TRK,(2,5),RLSE),
//                     UNIT=SYSDA,VOL=SER=XXXXXX,
//                     DISP=(,CATLG,DELETE),
//                     DCB=(RECFM=FB,BLKSIZE=8000,LRECL=80)
//SECTRE4      DD      DSN=STR.V310.FIX.INSTALL,SPACE=(TRK,(2,5),RLSE),
//                     UNIT=SYSDA,VOL=SER=XXXXXX,
//                     DISP=(,CATLG,DELETE),
//                     DCB=(RECFM=FB,BLKSIZE=8000,LRECL=80)
//SECTRE5      DD      DSN=STR.V310.RTMNTLD,SPACE=(CYL,(2,2),RLSE),
//                     UNIT=SYSDA,VOL=SER=XXXXXX,
//                     DISP=(,CATLG,DELETE),
//                     DCB=(RECFM=VB,BLKSIZE=2564,LRECL=256)
//*
//LOADPDS      EXEC    PGM=IEBCOPY,PARM=NEW
//SYSPRINT     DD      SYSOUT=*
//SRCEINDD     DSN=STR.V331.SOURCE,DISP=(OLD,PASS),
//                     UNIT=TAPE,VOL=SER=STR331,LABEL=(1,SL)
//SRCEOUT      DD      DSN=STR.V331.SOURCE,DISP=SHR
//LOADIN       DD      DSN=STR.V331.LOAD,DISP=(OLD,PASS),
//                     UNIT=TAPE,VOL=SER=STR331,LABEL=(2,SL)
//LOADOUT      DD      DSN=STR.V331.LOAD,DISP=SHR
//SYSUT3       DD      DSN=&&TEMP,UNIT=SYSDA,SPACE=(80,(60,45))
//SYSIN DD *
         COPY INDD=SRCEIN,OUTDD=SRCEOUT
         COPY INDD=LOADIN,OUTDD=LOADOUT
//*
//LOADRDME EXEC        PGM=IEBGENR
//SYSPRINT     DD      SYSOUT=*
//SYSUT1       DD      DSN=STR.V331.README,DISP=(OLD,PASS),
//                     UNIT=TAPE,VOL=SER=STR331,LABEL=(4,SL)
//SYSUT2       DD      DSN=STR.V331.README,DISP=SHR
//SYSIN        DD      DUMMY
//*
//LOADRDME EXEC        PGM=IEBGENR
//SYSPRINT     DD      SYSOUT=*
//SYSUT1       DD      DSN=STR.V331.FIX.INSTALL,DISP=(OLD,PASS),
//                     UNIT=TAPE,VOL=SER=STR331,LABEL=(5,SL)
//SYSUT2       DD      DSN=STR.V331.FIX.INSTALL,DISP=SHR
//SYSIN        DD      DUMMY
//*
//LOADNTLD EXEC        PGM=IEBGENR
//SYSPRINT     DD      SYSOUT=*
//SYSUT1       DD      DSN=STR.V331.RTMNTLD,DISP=(OLD,PASS),
//                     UNIT=TAPE,VOL=SER=STR331,LABEL=(3,SL)
//SYSUT2       DD      DSN=STR.V331.RTMNTLD,DISP=SHR
//SYSIN        DD      DUMMY
```

IX.4.2 Install zaps and fixes

This distribution of SECURITRE may have contained additional zaps and fixes that were created after the initial release date. Before continuing the installation, please refer to the STR.V331.README for a description of any zaps/fixes available. The STR.V331.FIX.INSTALL dataset contains information on installing these fixes.

IX.4.3 Assemble STRIOR Module

One of the source members included in the STR.V331.SOURCE PDS is STRIOR. This member contains all of the operating system and SSF dependent code in SECURITRE. This member must be assembled at each installation.

Prior to assembling the STRIOR module, a conditional assemble variable near the top of the STRIOR module must be modified to reflect the operating system in use and the installation options chosen for the ADARUN module. The two possible choices for ADARUN are:

```
&XA          SETC        'Y'          RUNNING ADARUN AMODE 31
&XA          SETC        'N'          RUNNING ADARUN AMODE 24
```

The following JCL, which is located in STR.V331.SOURCE(JCLASM), is provided as an example to assemble and link STRIOR:

```
//*     MEMBER(JCLASM)
//*     ASSEMBLE & LINK STRIOR, STRPARM, ETC.
//*
//TSIASM       PROC    MEM=U4GOT,
//                     SOURCE='STR.V331.SOURCE',
//                     LOAD='STR.V331.LOAD',
//                     SYSMAC='SYS1.MACLIB',
//                     SYSMAC2='STR.V331.SOURCE',
//                     SYSMAC3='ADABAS.SOURCE',
//                     LP=,
//                     UNIT='SYSDA'
//*
//ASSEM EXEC    PGM=ASMA00,REGION=2048K,
//      PARM='DECK,NOOBJECT,LIST,XREF(FULL),ALIGN'
//SYSLIB               DD      DISP=SHR,DSN=&SYSMAC
//                     DD      DISP=SHR,DSN=&SYSMAC2
//                     DD      DISP=SHR,DSN=&SYSMAC3
//                     DD      DISP=SHR,DSN=&SOURCE
//SYSUT1               DD      UNIT=&UNIT,SPACE=(1700,(400,400))
//SYSUT2               DD      UNIT=&UNIT,SPACE=(1700,(400,400))
//SYSUT3               DD      UNIT=&UNIT,SPACE=(1700,(400,400))
//SYSUT4               DD      UNIT=&UNIT,SPACE=(1700,(400,400))
//SYSPUNCH             DD      DSN=&&OBJMOD,UNIT=&UNIT,
//      DISP=(,PASS),SPACE=(400,(100,100)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT             DD      SYSOUT=*
//SYSIN        DD      DISP=SHR,DSN=&SOURCE(&MEM)
//* ...
//LNK   EXE CPGM=IEWL,PARM='LIST,XREF,LET,MAP,NCAL,AMODE=ANY,&LP',
//      COND=(3,LT,ASSEM)
//SYSLIB               DD      DISP=SHR,DSN=&LOAD
//SYSUT1               DD      UNIT=&UNIT,SPACE=(1024,(100,10))
//SYSLMOD              DD      DISP=SHR,DSN=&LOAD(&MEM)
//SYSPRINT             DD      SYSOUT=*
//SYSLIN               DD      DSN=&&OBJMOD,DISP=(OLD,DELETE)
//      PEND
//*
//STRIOR               EXEC    TSIASM,MEM=STRIOR
//*STNPNAT             EXEC    TSIASM,MEM=STNPNAT
//*STP00007            EXEC    TSIASM,MEM=STP00007,LOAD='ADABAS.LOAD'
//*TSIEX4PR            EXEC    TSIASM,MEM=TSIEX4PR
//
```

STRIOR and STPxxxxx must not be marked RENT or REUSE for the linkage editor. STRIOR is not reentrant or reusable. If STPxxxxx is marked RENT or REUSE, the RTM parameter reload will cause an ADABAS ABEND.

Note: Retain the assembler listing of STRIOR in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.4.4 Apply Authorization and Fix Zaps

You should have received an authorization Zap from TSI that will allow you to use SECURITRE. Apply it at this time.

IX.4.5 Link Edit User-Exit-1

Link-edit the individual User-Exit-1 components together with STRIOR to create STRUEX1. The following JCL is located in STR.V331.SOURCE(JCLLKED1):

```
//* MEMBER(JCLLKD1B) LINKEDIT STRUEX1
//*
//TSILKED      PROC    STRLOD='STR.V331.LOAD',
//                     LOAD='STR.V331.LOAD',
//                     LP=,
//                     UNIT='SYSDA'
//LNK   EXEC   PGM=IEWL,PARM='LIST,XREF,LET,MAP,NCAL,AMODE=ANY,&LP'
//SYSUT1       DD      UNIT=&UNIT,SPACE=(1024,(100,10))
//STRLOD       DD      DISP=SHR,DSN=&STRLOD
//SYSLMOD      DD      DISP=SHR,DSN=&LOAD
//SYSPRINT     DD      SYSOUT=*
//SYSLIN       DD      DDNAME=CARDS
//       PEND
//* ------------------------------------------------------------------
//*
//* LKED USER EXIT 1
//*
//UEX1P2 EXEC TSILKED,LOAD='ADABAS.LOAD'
//CARDS DD *
  INCLUDE      STRLOD(STRMAIN)        MAINLINE DRIVER
  INCLUDE      STRLOD(STRFLSP)        FLS PROCESSING
  INCLUDE      STRLOD(STRDSNB)        BUILD PSEUDO-DSN
  INCLUDE      STRLOD(STRSUB1)        COMMON SUBROUTINES
  INCLUDE      STRLOD(STRSUB2)        USER TABLE SUBROUTINES
  INCLUDE      STRLOD(STRSUB3)        FLS TABLE SUBROUTINES
  INCLUDE      STRLOD(STRTAB) TABLE HANDLE
  INCLUDE      STRLOD(STRTAB2)        DSN TABLE HANDLER
  INCLUDE      STRLOD(STRINIT)        INITIALIZATION
  INCLUDE      STRLOD(STRINTV)        INTERVAL HANDLE
  INCLUDE      STRLOD(STRERR) ERROR HANDLE
  INCLUDE      STRLOD(STRFFNR)        BINARY SEARCH FILE LOOKUP
  INCLUDE      STRLOD(STRMON) HANDLE MONITOR REQUESTS
  INCLUDE      STRLOD(STRPUPD)        PARAMETER UPDATE
  INCLUDE      STRLOD(STRTRAC)        TRACE ROUTINE
  INCLUDE      STRLOD(STRTRCI)        TRACE ROUTINE
  INCLUDE      STRLOD(STRDATA)        DATA
  INCLUDE      STRLOD(STRZAP) ZAP SETTINGS
  INCLUDE      STRLOD(STRABEND)       ABEND MESSAGES
  INCLUDE      STRLOD(STRIOR)
  MODE  RMODE(24),AMODE(ANY)
  NAME  STRUEX1(R)
//
```

Note: DO NOT CHANGE THE ORDER OF THE INCLUDE CARDS.

SECURITRE User-Exit-1 must not be marked RENT or REUSE for the linkage editor. SECURITRE User-Exit-1 is not reentrant or reusable.

Note: Retain the Linkage Editor map listing in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.4.6 Code "STRPARM" Module

The "STRPARM" module is provided to allow the Security Administrator to tailor SECURITRE User-Exit-1 to site-specific requirements. The "STRPARM" module is also used by the SECURITRE ADARUN front-end module used for ADABAS Utility Control. A sample "STRPARM" module is provided in STR.V331.SOURCE(STP00007). For more information about the parameter statements used, refer to **Section II - Setting Up SECURITRE for ADABAS**.

There must be one "STRPARM" module for each database for which SECURITRE is active. The module name must conform to the following convention:

```
STP99999
```

In this case, 99999 is the DBID for that database. For example, the "STRPARM" module name for DBID=00007 is STP00007.

IX.4.7 Assemble STRPARM Module

To assemble the "STRPARM" parameter module, the following JCL is provided as an example. However, it is recommended that installation personnel customize existing JCL to accomplish this step.

Note: When using the same JCL as in **Section X.4.3 - Assemble STRIOR Module**, uncomment the STRPARM assembly as shown here:

```
//*
//*STRIOR      EXEC    TSIASM,MEM=STRIOR
//*STNPNAT     EXEC    TSIASM,MEM=STNPNAT
//STP00251     EXEC    TSIASM,MEM=STP00251,LOAD='ADABAS.LOAD'
//*TSIEX4PR    EXEC    TSIASM,MEM=TSIEX4PR
//
```

STPxxxxx must not be marked RENT or REUSE for the linkage editor because when using the RTM parameter reload, an ADABAS abend will occur.

Note: Retain the assembler listing in a safe place. TSI technical support personnel may require information from this listing to help resolve any problems that may arise.

IX.4.8 APF-Authorization

Because SECURITRE issues privileged SVCs to validate a user's ability to access a given file, the ADABAS and SECURITRE modules must reside in an APF-authorized library. This means that all of the loadlibs in the steplib for the ADABAS nucleus must be APF-authorized. If one loadlib is not APF-authorized, then all APF-authorization is lost. It is acceptable for modules to be loaded from a loadlib located in the "link list" if that library is APF-authorized. With this in mind, both the ADABAS loadlib and the SECURITRE loadlib must be APF-authorized, and ADARUN must be re-linked with SETCODE AC(1) to make it APF authorized as well.

The following sample JCL is located in STR.V331.SOURCE(JCLLKDA) will allow you to APF authorize STRU:

```
//*
//*     APF-AUTHORIZATION
//*
//LKED  EXEC PGM=IEWL,PARM='MAP,LIST,NCAL'
//SYSLMOD      DD      DSN=ADABAS.LOAD,DISP=SHR
//ADALIB       DD      DSN=ADABAS.LOAD,DISP=SHR
//SYSUT1       DD      UNIT=SYSDA,SPACE=(CYL,(1,2))
//SYSPRINT     DD      SYSOUT=*
//SYSLIN       DD      *
  INCLUDE      ADALIB(ADARUN)
  SETCODE      AC(1)
  NAME  ADARUN(R)
//
```

IX.4.9 Install TSIRDC

Note:	If NATURAL is not running under CICS skip to the section IX.4.10.

IX.4.9.1 Modify the CICS TWA

Note:	If SECURITRE and TRIM are installed together, this step only needs to be completed once for both products. The same value will be used for both products when applying the TWA offset zap to STRUEXB3 and TRMUEXB2.

Modify the CICS PCT entry for the NATURAL transaction to increase the TWASIZE by four bytes. For example, if the default of 128 byes is used increase it to 132 bytes.

Note:	You will need the original TWASIZE value for use in the next step.

IX.4.9.2 Apply the TWA offset ZAP

Note: This step is only necessary if the original TWASIZE from step IX.4.9.1 is different than the default of 128.

Apply the TWA offset zap to STRUEXB3 module. This zap can also be found in TSI.STR.V331.SOURCE(JCLTWAOF)

xxxx should be replaced with the hexadecimal representation of the original TWA value. For example: If the original TWA value was 64, xxxx would be replaced with 0040.

```
NAME STRUEXB3 STRUEXB3
VER 02BA 0080
REP 02BA xxxx
NAME TSIRDC31 TSIRDC
VER 0102 0080
REP 0102 XXXX
NAME TSIRDC41 TSIRDC
VER 0102 0080
REP 0102 XXXX
NAME TSIRDC42 TSIRDC
VER 00FA 0080
REP 00FA XXXX
```

Note: If multiple CICS transactions for NATURAL are used and each have different values for TWA, a separate STRUEXB3 and NATURAL nuclei (containing separate TSIRDC modules) will need to be used for each environment or the CICS TWA sizes must be made consistent across environments.

IX.4.9.3 Relink the NATURAL nucleus with TSIRDC included

The NATURAL nucleus must be re-linked in one of the ways described below.

The installation personnel may make modifications to the existing NATURAL link JCL. A DDNAME of STRLIB must be added to point to STR.V331.LOAD.

Insert all the INCLUDE cards before 'INCLUDE NATLIB(NATPM)'.

NATURAL 4.2

The following INCLUDE card is used for NATURAL 4.2 only.

```
INCLUDE STRLIB(TSIRDC42)
```

NATURAL 4.1

The following INCLUDE card is used for NATURAL 4.1 only.

```
INCLUDE STRLIB(TSIRDC41)
```

NATURAL 3.1.6

The following INCLUDE card is used for NATURAL 3.1.6 only:

```
INCLUDE STRLIB(TSIRDC31)
```

IX.4.10 ADABAS Link Routine User-Exit B

Note:	If SECURITRE is to co-exist with APAS, skip this section and refer to **Appendix B**.

With ADABAS, it is possible to specify user-exits to ADALNK (Batch and TSO), LNKOLM LNKOL (CICS), and TLOPADAB (COM-PLETE). These exits are UEXITB (given control prior to the call to ADABAS) and UEXITA (given control after the call has completed).

SECURITRE makes use of User-Exit-B to gather information about the user's environment and to pass this information to STRUEX1. The SECURITRE supplied UEXITB must be installed in order to obtain the User-ID if the USERID=STRUEXB option was selected in the STRPARMS. Refer to **Section IX.4.6 - Code "STRPARM" module**.

IX.4.10.1 Batch and TSO

To install the SECURITRE User-Exit-B under BATCH and TSO, follow these steps:

- Near the top of ADALNK in the ADABAS source library, modify the following equate accordingly:

  If SECURITRE is to run without TRIM, LNUINFO must be set to a value of at least 176 to allow SECURITRE User-Exit-B to pass 176 bytes of information about each command to STRUEX1.

  If SECURITRE and TRIM are to run together, LNUINFO must be set to a value of at least 416 to allow TRIM and SECURITRE User-Exit-Bs to pass information about each user to their User-Exit-1s and 4s. For more information about LNUINFO, refer to the ADABAS DBA Reference Manual and the ADABAS Implementation and Maintenance Manual.

- Reassemble ADALNK. (JCLASM should be used.)

- Relink Adalink.

  If SECURITRE is to run without TRIM, include the SECURITRE User-Exit-B (STRUEXBB) within the ADALNK load module using the Linkage Editor. The Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(ADALNK) VANILLA BATCH LINK ROUTINE
INCLUDE STRLIB(STRUEXBB)      SECURITRE BATCH USER-EXIT-B
NAME ADALNK(R)
```

  If SECURITRE and TRIM are to run together, the Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(ADALNK) VANILLA BATCH LINK ROUTINE
CHANGE  UEXITBU(TRMUEXBB)     STR CALLS TRIM BATCH USER-EXIT-B
INCLUDE STRLIB(STRUEXBB)      SECURITRE BATCH USER-EXIT-B
CHANGE  UEXITB(TRMNOENT)      RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXBB)      TRIM BATCH USER-EXIT-B
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR NON-CICS
INCLUDE USRLIB(UEXIBUB)       SITE USER-EXIT-A FOR BATCH (OPTIONAL)
NAME ADALNK(R)
```

- A user supplied User-Exit-B may also be included as part of the link-edit process. For more information about interfacing SECURITRE User-Exit-B with an installation written User-Exit-B, refer to **Section IX.4.10.4 - User-Supplied User-Exit-B**.

- If "ADABAS Migration Aids" is being used, replace all references to ADALNK with ADALNKx where x is the ADABAS version.

IX.4.10.2 CICS

Follow these steps to install the SECURITRE User-Exit-B under CICS:

- Near the TOP of ADAGSET in the ADABAS source library, modify the following macro global variables accordingly:

  If SECURITRE is to run without TRIM, LUINFO must be set to a value of at least 176 to allow SECURITRE User-Exit-B to pass 176 bytes of information about each command to STRUEX1, and LUSAVE must also be set to a value of at least 72.

  If SECURITRE and TRIM are to run together, LUINFO must be set to a value of at least 416 to allow TRIM and SECURITRE User-Exit-Bs to pass information about each command to their User-Exit-1s and -4s, and LUSAVE must also be set to a value of 72.

- Reassemble LNKOLM and LNKOLSC:

  If SECURITRE is to run without TRIM, include the SECURITRE User-Exit-B (STRUEXB3) within the ADABAS load module using the linkage editor. The Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(LNKOLM)
INCLUDE ADALIB(LNKOLSC)
INCLUDE STRLIB(STRUEXB3)      SECURITRE CICS UEXB
NAME ADABAS (R)
```

  If SECURITRE and TRIM are run together, the Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(LNKOLM)
INCLUDE ADALIB(LNKOLSC)
CHANGE  UEXITBU(TRMUEXB2)     STR CALLS TRIM CICS UEXB
INCLUDE STRLIB(STRUEXB3)      SECURITRE CICS UEXB
CHANGE  UEXITB(TRMNOENT)      RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXB2)      TRIM CICS USER-EXIT-B
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR BATCH
INCLUDE USRLIB(UEXIBUB)       SITE USER-EXIT-A FOR BATCH (OPTIONAL)
NAME ADABAS (R)
```

- A user supplied User-Exit-B may also be included as part of the link-edit process. For more information, refer to **Section IX.4.9.4 - User-Supplied User-Exit-B**.

- The CICS region **must** then be recycled to allow for installation of the resident module ADABAS.

IX.4.10.3 COM-PLETE

To install the SECURITRE User-Exit-B under COM-PLETE/TPF version 4.5 and above, follow these steps:

- Near the top of ADALCO, modify the following equate accordingly:

  If SECURITRE is to run without TRIM, LNUINFO must be set to a value of at least 176 to allow SECURITRE User-Exit-B to pass 176 bytes of information about each command to STRUEX1.

  If SECURITRE and TRIM are run together, LNUINFO must be set to a value of at least 416 to allow TRIM and SECURITRE User-Exit-Bs to pass information about each user to their User-Exits-1s and 4s. For more information about LNUINFO, refer to the ADABAS DBA Reference Manual and the ADABAS Implementation and Maintenance Manual.

- Reassemble ADALCO.

  If SECURITRE is to run without TRIM, include the SECURITRE User-Exit-B (STRUEXB5) in the ADALCO load module using the linkage editor. The Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(ADALCO)        COM-PLETE LINK ROUTINE
INCLUDE STRLIB(STRUEXB5)             SECURITRE COM-PLETE UEXB
NAME ADALCO(R)
```

  If SECURITRE and TRIM are run together, the Linkage Editor control cards should appear as follows:

```
INCLUDE ADALIB(ADALCO)        COM-PLETE LINK ROUTINE
CHANGE  UEXITBU(TRMUEXB5)            STR CALLS TRIM UEXB5
INCLUDE STRLIB(STRUEXB5)             SECURITRE COM-PLETE UEXB
CHANGE  UEXITB(TRMNOENT)             RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXB5)             TRIM COM-PLETE UEXB
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR COM-PLETE
INCLUDE USRLIB(UEXIBUB)       SITE USER-EXIT-A FOR COM-PLETE (OPTIONAL)
NAME ADALCO(R)
```

- A user-supplied User-Exit-B may also be included as part of the link-edit process. For more information about interfacing SECURITRE User-Exit-B with an installation-written User-Exit-B, refer to **Section IX.4.9.4 - User-Supplied User-Exit-B**.

IX.4.10.4 User-Supplied User-Exit-B

A user-supplied User-Exit-B may also be included as part of the linkage edit process. To interface the SECURITRE User-Exit-B with an installation-written User-Exit-B, the user must supply a CSECT name or Entry name of UEXITBU in their user-exit.

The registers on entry to UEXITBU will be the same as published by Software AG for entry to UEXITB. In addition, the SECURITRE USERINFO Area will be provided to UEXITBU.

CO-EXISTENCE Without TRIM
If SECURITRE is to run without TRIM, the SECURITRE User-Exit-B will be invoked by ADALNK/ADALNC and will, in turn, invoke the user-supplied User-Exit-B before returning to ADALNK/ADALNC. The registers on entry to UEXITBU will be the same as published by Software AG for entry to UEXITB. In addition, the SECURITRE USERINFO Area will be provided to UEXITBU.

- For BATCH/TSO, re-link ADALNK using the following Linkage Editor control cards:

```
INCLUDE ADALIB(ADALNK)  BATCH LINK ROUTINE
INCLUDE STRLIB(STRUEXBB) SECURITRE BATCH USER-EXIT-B
INCLUDE USRLIB(UEXITBU)        USER'S USER-EXIT-B
NAME ADALNK(R)
```

- For CICS, re-link LNKOLM & LNKOLSC using the following Linkage Editor control cards:

```
INCLUDE ADALIB(LNKOLM)
INCLUDE ADALIB(LNKOLSC)
INCLUDE STRLIB(STRUEXB)       SECURITRE CICS USER-EXIT-B
INCLUDE USRLIB(UEXITBU)       USER'S USER-EXIT-B
NAME ADABAS(R)
```

- For COM-PLETE or TPF 4.5+ re-link ADALCO using the following Linkage Editor control cards:

```
INCLUDE ADALIB(ADALCO)   COM-PLETE LINK RTNE
INCLUDE STRLIB(STRUEXB5) SECURITRE COM-PLETE USER-EXIT-B
INCLUDE USRLIB(UEXITBU)  USER'S USER-EXIT-B
NAME ADALCO (R)
```

CO-EXISTENCE With TRIM

If SECURITRE and TRIM are to run together, the SECURITRE User-Exit-B is designed to call entry UEXITBU when it is finished processing. By changing the reference of UEXITBU to TRMUEXBB (or TRMUEXB), SECURITRE may call the TRIM User-Exit-B that will invoke the user-supplied User-Exit-B that has an entry point of UEXITBU before returning to ADALNK/ADALNC.

- For BATCH/TSO, re-link ADALNK using the following Linkage Editor control cards:

```
INCLUDE ADALIB(ADALNK)   BATCH LINK ROUTINE
CHANGE  UEXITBU(TRMUEXBB) STR CALLS TRIM BATCH USER-EXIT-B
INCLUDE STRLIB(STRUEXBB)  SECURITRE BATCH USER-EXIT-B
CHANGE  UEXITB(TRMNOENT)  RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXBB)  TRIM BATCH USER-EXIT-B
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR NON-CICS
INCLUDE USRLIB(UEXITBU)         USER'S USER-EXIT-B UEXITBU ENTRY
NAME ADALNK(R)
```

- For CICS, re-link LNKOLM and LNKOLSC using the following Linkage Editor control cards:

```
INCLUDE ADALIB(LNKOLM)        PART OF CICS LINK ROUTINE
INCLUDE ADALIB(LNKOLSC)       PART OF CICS LINK ROUTINE
CHANGE  UEXITBU(TRMUEXB2)     STR CALLS TRIM CICS USER-EXIT-B
INCLUDE STRLIB(STRUEXB3)      STR USER-EXIT-B FOR CICS
CHANGE  UEXITB(TRMNOENT)      RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXB2)      TRIM USER-EXIT-B FOR CICS
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR BATCH
INCLUDE USRLIB(UEXITBU)       SITE USER-EXIT-B FOR CICS (OPTIONAL)
NAME ADABAS(R)
```

- For COM-PLETE or TPF 4.5+ re-link ADALCO using the following Linkage Editor control cards:

```
INCLUDE ADALIB(ADALCO)   COM-PLETE LINK ROUTINE
CHANGE  UEXITBU(TRMUEXBB) STR CALLS TRIM COM-PLETE USER-EXIT-B
INCLUDE STRLIB(STRUEXBB)  SECURITRE COM-PLETE USER-EXIT-B
CHANGE  UEXITB(TRMNOENT)  RENAME TRIM UEXITB ENTRY
INCLUDE TRMLIB(TRMUEXBB)  TRIM COM-PLETE USER-EXIT-B
INCLUDE TRMLIB(TRMUEXA)       TRIM USER-EXIT-A FOR COM-PLETE
INCLUDE USRLIB(UEXITBU)         USER'S USER-EXIT-B UEXITBUENTRY
NAME ADALCO(R)
```

The SECURITRE USERINFO Area is located at R1+64 upon entry to UEXITBU and is made available to UEXITBU is as follows:

```
0000       USERINFO      DS     0H
0000       UILEN         DC     AL2(UIEND-UILEN)      LENGTH OF SEGMENT
0002       UISEG         DC     CL4'STRE'             SEGMENT IDENTIFIER
0007       UIVERNO       DC     CL4'V331'             CURR.VERSION/RELEASE
000A       UIFACIL       DC     CL4' '                FACILITY (CICS,TSO,ETC)
000E       UIENVI        DS     XL14                  ENVIRONMENT INFO
001C       UITIMEST      DC     XL4'0'                COMMAND START STCK
0020       UISTRADR      DC     XL4'0'                ADDRESS IN STR TABLES
           *
0024       UIGENJOB      DC     CL8' '                GENERAL JOBNAME
002C       UIGENSTP      DC     CL8' '                GENERAL STEPNAME
0034       UIGENPGM      DC     CL8' '                GENERAL PROGRAM NAME
003C       UIGENJID      DC     CL4' '                GENERAL JES NUMBER
           *
0040       UINATPGM      DC     CL8' '                NATURAL PROGRAM
0048       UINATAPL      DC     CL8' '                NATURAL APPLICATION
0050       UINATUID      DC     CL8' '                NATURAL USERID
0058       UINATFUS      DC     XL2'0000'             NATURAL FUSER FNR
           *
005A       UICOMTID      DC     CL4' '                COM-PLETE TID
005E       UIUSRID       DC     CL8' '                SECURITY ACC USERID
0066       UIGRP         DC     CL8' '                SECURITY GROUP ID
006E       UINODE        DC     CL4' '                NODE (SMFID)
0072       UITABL        DC     CL4' '                TABLE ENTRY ADDRESS
           *
           UICICTRM      EQU    UICOMTID              CICS TERMID
           UICICUID      EQU    UIUSRID               CICS USERID
0076       UICICTRN      DC     CL4' '                CICS TRANID
007A       UICICOPI      DC     CL3' '                CICS OPID
007D       UICICTSK      DC     CL5' '                CICS TASKID
           *
0082       UINATDB       DC     XL2'0000'             NATURAL FUSER DBID
           *
0088       UIEND         DS     0D
           UITXT         EQU    UILEN+6
           UI            EQU    UILEN
```

Figure 15 – USERINFO Area Format

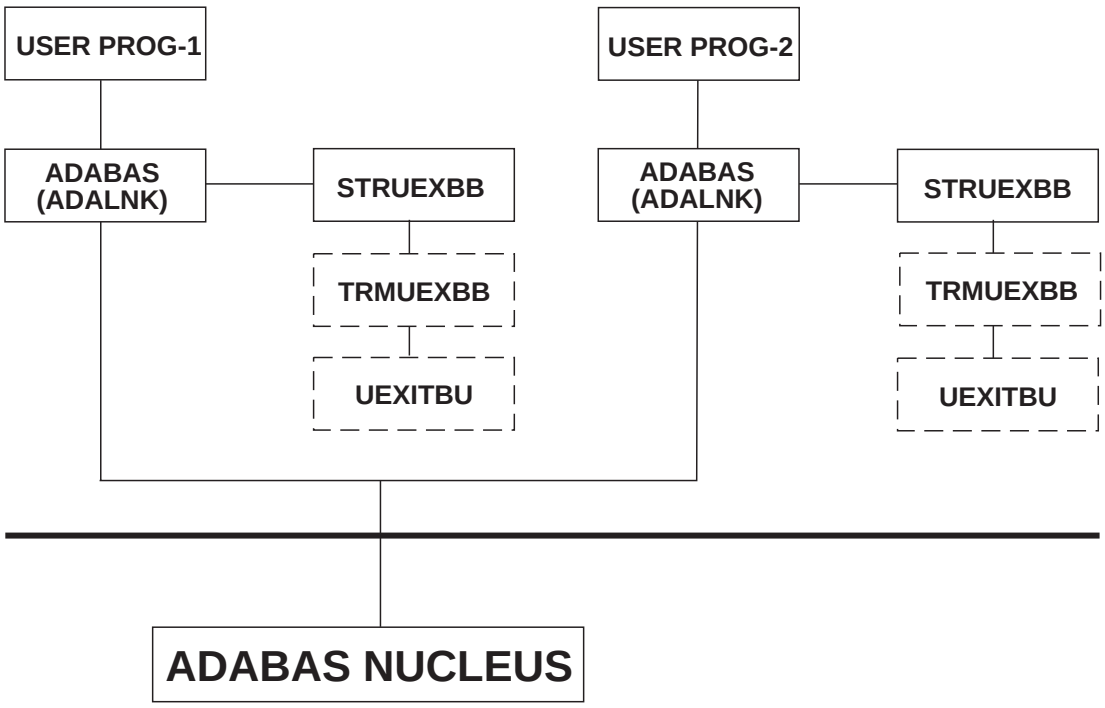

Figure 16 – ADABAS Batch/TSO Environment

In an ADABAS BATCH/TSO environment, there may be one or more user programs that communicate directly with the ADABAS nucleus. To communicate with ADABAS, these programs use the ADALNK link routine. The link routine will pass control to STRUEXBB before going to ADABAS. STRUEXBB places the User-ID in the USERINFO area and passes control to the user's User-Exit-B (UEXITBU) for any additional processing. UEXITBU returns to STRUEXBB, which returns to ADALNK, which in turn passes the command to the ADABAS nucleus for processing.

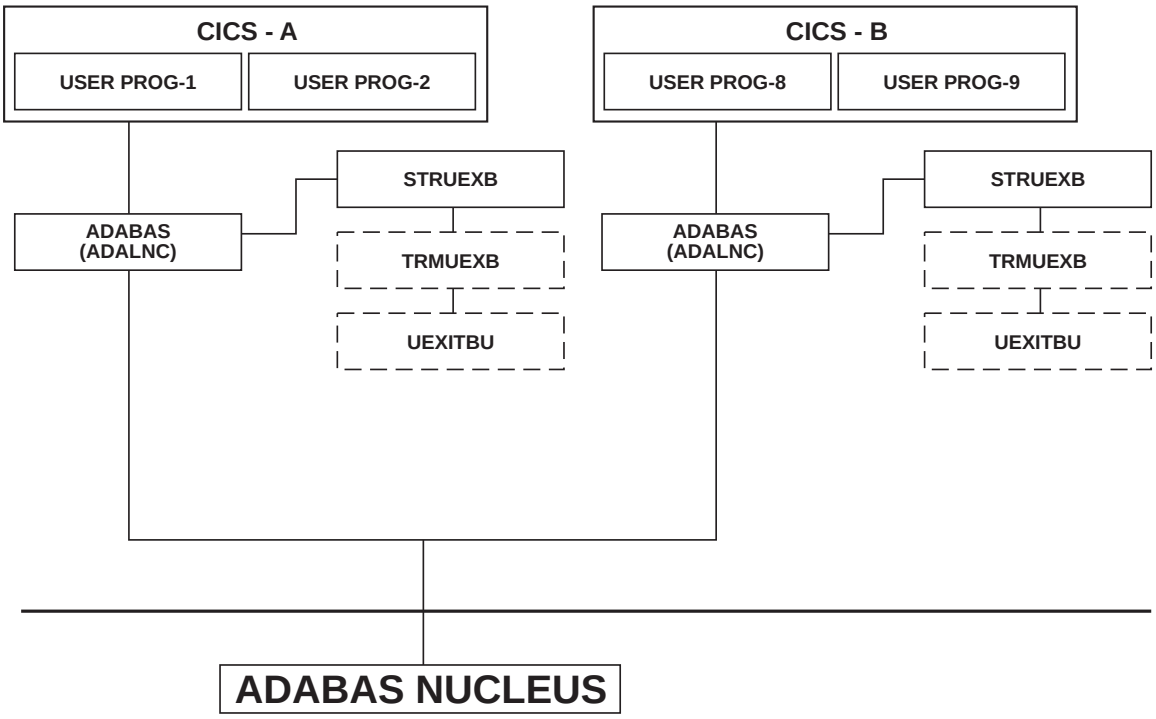

Figure 17 – ADABAS CICS Environment

In an ADABAS CICS environment, there may be one or more CICS regions, in which one or more user programs may be running. To communicate with ADABAS, CICS uses the ADALNC link routine. The link routine will pass control to STRUEXB before going to ADABAS. STRUEXB places the User-ID in the USERINFO area and passes control to the user's User-Exit-B (UEXITBU) for any additional processing. UEXITBU returns to STRUEXB, which returns to ADALNC, which in turn passes the command to the ADABAS nucleus for processing. ADABAS returns the result of the command to ADALNC, which returns the result of the command to the user.

IX.4.11 Activate SECURITRE for ADABAS

SECURITRE User-Exit-1 and User-Exit-4 must be in operation with ADABAS in order for SECURITRE to function. User-Exit-1 performs the validation through the SSF and is the "core" of the product.

The SECURITRE User-Exit-4 (STRUEX4) is required to allow operation of the SECURITRE RTM and to provide the proper ADABAS response code (200) to the calling program in the event that the SECURITRE User-Exit-1 has failed the command.

If SECURITRE and TRIM are to run together and no user's User-Exit-4 is desired, TRIM's User-Exit-4 will do all of the processing that is required for SECURITRE.

If there is already an existing User-Exit-4, STRUEX4 may co-exist with this User-Exit. For more information about allowing multiple User-Exit-4 programs to co-exist, refer to **Section VIII - User-Exit Co-Existence**.

To activate SECURITRE ADABAS file level security, the following changes must be made to each ADABAS nucleus under which SECURITRE will execute:

- The following parameters must be specified as part of the ADARUN parameters if SECURITRE's User-Exit-4 is the only User-Exit-4 being executed:

```
ADARUN UEX1=STRUEX1
ADARUN UEX4=STRUEX4
ADARUN LOGGING=YES
```

- The following parameters must be specified as part of the ADARUN parameters if SECURITRE and TRIM are running together and there is no User-Exit-4 being executed:

```
ADARUN UEX1=STRUEX1
ADARUN UEX4=xxxxxxx
(xxxxxxx is the name of the TRIM User-Exit-4 being used)
ADARUM LOGGING=YES
```

- The following parameters must be specified as part of the ADARUN parameters if any user's User-Exit-4 is to co-exist with SECURITRE (refer to **Section VIII.2 – User-Exit-4 Co-Existence** for additional information):

```
ADARUN UEX1=STRUEX1
ADARUN UEX4=TSIUEX4
ADARUN LOGGING=YES
```

- The Region Size for the ADABAS nucleus must be adequate for SECURITRE to operate. **The SECURITRE region size requirements are given in the Storage Requirements section of this manual.**

- In addition, STRUEX4 requires that a valid (non-dummy) Command Log dataset be specified in the ADABAS start-up JCL. STRUEX4 will not cause any records to be logged to the command log. The specification of this dataset is to satisfy a requirement of ADABAS (i.e., ADABAS will not pass control to User-Exit-4 unless logging is anticipated). Either a Single Command Log or Dual Command Logs may be specified.

- The following JCL statement must be added to the ADABAS nucleus startup JCL:

```
//STRMSG DD SYSOUT=*
```

  This statement allows various informational messages to be printed. For more information, refer to **Section X.5 - STRMSG Messages**.

- The ADABAS nucleus must be restarted to allow loading of the SECURITRE User-Exit-1 and User-Exit-4.

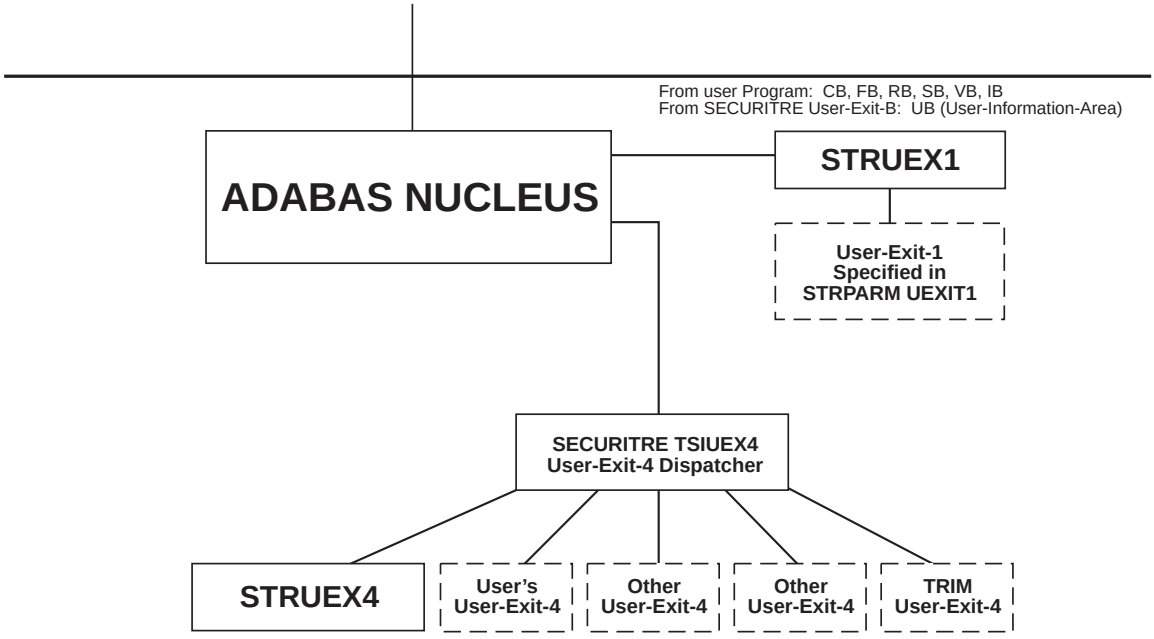

Figure 18 – ADABAS 5 User-Exit Environment

When ADABAS receives a command, it passes control to SECURITRE User-Exit-1 (STRUEX1). SECURITRE User-Exit-1 verifies the user's authority to access the requested data and passes control to the user's User-Exit-1 for any additional checking. (Control may also be passed to any other User-Exit-1 in use, such as TRIM User-Exit-1.) After verification, the command is returned to ADABAS for processing. Once the command has been processed, ADABAS passes control to the SECURITRE User-Exit-4 "Dispatcher" (TSIUEX4), which passes control to SECURITRE User-Exit-4 (STRUEX4), and then to any other User-Exit-4s in turn. After all User-Exit-4 processing is complete, control is passed to ADABAS, which returns the result of the command to the link routine, which returns the results to the user/program issuing the command.

IX.4.12 Co-existing User-Exits

If co-existing User-Exit-4(s) is (are) necessary in the site's ADABAS environment, then the following subsections must be completed.

IX.4.12.1 Assemble TSIEX4PR Module

One of the source members included in the STR.V331.SOURCE PDS is TSIEX4PR. The member contains a sample TSIEX4P parameter module. This member must be modified as required (Refer to **Section VIII.2 - User-Exit-4 Co-existence**) and assembled at each installation.

```
//*
//*STRIOR      EXEC    TSIASM,MEM=STRIOR
//*STNPNAT     EXEC    TSIASM,MEM=STNPNAT
//*STP00007    EXEC    TSIASM,MEM=STP00007,LOAD='ADABAS.LOAD'
//TSIEX4PR     EXEC    TSIASM,MEM=TSIEX4PR
//
```

Note: Retain the assembler listing of TSIEX4PR in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.4.12.2 Link-edit TSIUEX4 Components

It is necessary to link the TSIEX4PR module created in the previous step with the TSIEX4DR OBJECT module on the release tape to create a TSIUEX4 module that will be used as User-Exit-4 to the ADABAS nucleus. The following JCL is located in STR.V331.SOURCE(JCLLKED4):

```
//*    MEMBER(JCLLKED4) LINKEDIT TSIEX4DR USER EXIT 4
DISPATCHER
//*
//LKED  EXEC    PGM=IEWL,PARM='MAP,LIST,NCAL'
//SYSLIN        DD      DDNAME=CARDS
//SYSUT1        DD      UNIT=SYSDA,SPACE=(CYL,(1,2))
//SYSLMOD       DD      DSN=ADABAS.LOAD,DISP=SHR
//STROBJ        DD      DSN=STR.V331.LOAD,DISP=SHR
//STRLOD        DD      DSN=STR.V331.LOAD,DISP=SHR
//SYSPRINT      DD      SYSOUT=*
//CARD  DD *
  INCLUDE       STROBJ(TSIEX4DR)        DISPATCHER
  INCLUDE       STRLOD(TSIEX4PR)        PARMS
  NAME  TSIUEX4(R)                      REPLACE DISPATCHER
//
```

Note: Retain the Linkage Editor map listing in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.5 Install ADABAS Utility Security

ADABAS Utility Security is an optional component of SECURITRE. The following subsections describe how SECURITRE must be integrated with ADABAS if you wish to use ADABAS Utility Security. In general, this process consists of tailoring SECURITRE to site specifications, link-editing individual load modules, applying any authorization and fix Zaps, link-editing appropriate modules to the ADABAS load libraries, and activating SECURITRE.

The following steps are necessary only if ADABAS Utility Security is being used:

U1 Apply the authorization Zap and any fix Zaps to individual load modules. (Refer to section IX.5.1.)

U2 Link-edit individual load modules to create STURUNL. (Refer to section IX.5.2.)

U3 Link-edit to front-end ADARUN. (Refer to section IX.5.3.)

U4 Modify the "STRPARM" parameter module with the desired ADABAS Utility Security parameter settings. Before coding an "STRPARM" module refer to **Section IV - SECURITRE for ADABAS Utilities** for examples of uses of this parameter module relating to ADABAS Utility Security. (Refer to section IX.5.4 for installation instructions.)

U5 Assemble and link-edit the "STRPARM" parameter module into the load library. (Refer to section IX.5.5.)

U6 APF-authorize ADABAS and SECURITRE ADABAS Utility Security modules. (Refer to section IX.5.6.)

Repeat steps U3 through U6 for each ADARUN where SECURITRE ADABAS Utility Security is desired.

Use the following checklist for the **optional** installation of SECURITRE's ADABAS Utility Security under ADABAS V5 and V6:

✓	**STEP**	**FUNCTION**	**SECTION**	**REQUIRED**
	U1	Apply the authorization Zap and any fix Zaps	IX.5.1	Y
	U2	Link-edit STURUNL	IX.5.2	Y
	U3	Link-edit ADARUN front-end	IX.5.3	Y
	U4	Modify STRPARM	IX.5.4	Y
	U5	Assemble/Link STRPARM	IX.5.5	Y
	U6	APF-Authorize	IX.5.6	Y

Repeat steps U3 through U6 for other ADARUNs.

IX.5.1 Apply Authorization Zap and Fix Zaps

You should have received an authorization Zap from TSI that will allow the use of SECURITRE. Apply this Zap at this time. If you have received any fix Zaps, apply them at this time.

IX.5.2 Link-edit Individual Modules to Create STURUNL

Link-edit the individual utility security components together with STRIOR to create STURUNL. The following JCL is located in STR.V331.SOURCE(JCLLKDU1):

```
//*     MEMBER(JCLLKDU1) LINKEDIT UTILITY SECURITY STURUNL
//*
//TSILKED      PROC    LOAD='STR.V331.LOAD',
//                     LP=,
//                     UNIT='SYSDA'
//LNK   EXEC    PGM=IEWL,PARM='LIST,XREF,LET,MAP,NCAL,AMODE=ANY,&LP'
//SYSUT1       DD      UNIT=&UNIT,SPACE=(1024,(100,10))
//STRLOD       DD      DISP=SHR,DSN=&LOAD
//SYSLMOD      DD      DISP=SHR,DSN=&LOAD
//SYSPRINT     DD      SYSOUT=*
//SYSLIN       DD      DDNAME=CARDS
// PEND
//* --------------------------------------------------------------------
//*
//*     LKED COMPONENTS TOGETHER TO CREATE STURUNL
//*
//UTILP2       EXEC    TSILKED
//CARDS  DD *
  INCLUDE      STRLOD(STURUN)     MAINLINE DRIVER
  INCLUDE      STRLOD(STUSTR)     STRING HANDLE
  INCLUDE      STRLOD(STUDATA)    DATA
  INCLUDE      STRLOD(STUTAB7)    REPLACE WITH STUTAB6 FOR ADABASE version 6
  INCLUDE      STRLOD(STUOPER)    OPERCOM TABLE
  INCLUDE      STRLOD(STUINIT)    INITIALIZATION
  INCLUDE      STRLOD(STUFAIL)    ERROR HANDLE
  INCLUDE      STRLOD(STUZAP)     ZAP STATUS
  INCLUDE      STRLOD(STRIOR)     OPERATING SYSTEM DEPENDENT CODE
  INCLUDE      STRLOD(STUFFNR)    BINARY TABLE SEARCH MODULE
  NAME  STURUNL(R)
//* --------------------------------------------------------------------
//
```

Note: DO NOT CHANGE THE ORDER OF THE INCLUDE CARDS.

A condition code of 4 is normal for all of the preceding link-edit steps. The warning message IEW0461 RUNADA will appear. The module RUNADA will be resolved by the link-edit in the next step. The warning message IEW2454W 9203 SYMBOL ZSTRFFNR UNRESOLVED will appear and should be ignored.

Note: Retain the Linkage Editor map listing of STURUNL in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.5.3 Link-edit STURUNL and ADARUN

Note: You may receive IEHxxxI messages, they may be ignored.

It is necessary to link-edit the ADARUN module together with the STURUNL load module produced in the previous step to create a new ADARUN module that will be used to check ADABAS Utility Security. The following JCL is located in STR.V331.SOURCE(JCLLKDU2):

```
//*            MEMBER(JCLLKDU2)
//*            LINKEDIT TO REPLACE ADARUN
//*
//*            NOTE: WHEN THIS JCL IS USED WITH ADABAS VERSIONS
//*            THE CARD WHICH READS CHANGE RUNMVS(RUNADA)
//*            MUST BE CHANGED TO CHANGE ADARUN(RUNADA)
//*
//LKED  EXEC   PGM=IEWL,PARM='MAP,LIST,NCAL'
//SYSLIN       DD     DDNAME=CARDS
//SYSUT1       DD     UNIT=SYSDA,SPACE=(CYL,(1,2))
//SYSLMOD      DD     DSN=ADABAS.LOAD,DISP=SHR
//ADALOD       DD     DSN=ADABAS.VANILLA.LOAD,DISP=SHR
//STRLOD       DD     DSN=STR.V331.LOAD,DISP=SHR
//SYSPRINT     DD     SYSOUT=*
//CARDS DD *
  INCLUDE STRLOD(STURUNL)                    MAINLINE DRIVER
  CHANGE RUNMVS(RUNADA)                      FLIP CSECT NAME
  INCLUDE ADALOD(ADARUN)                     VANILLA ADARUN
  MODE RMODE(24),AMODE(ANY)                  SET AMODE/RMODE
  NAME ADARUN(R)                             REPLACE ADARUN
//
```

Note: DO NOT CHANGE THE ORDER OF THE INCLUDE CARDS.

For versions of ADABAS prior to 5.2, change the line that reads CHANGE RUNMVS(RUNADA).

Note: Retain the Linkage Editor map listing in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

For ADABAS Utility Control, the SECURITRE ADARUN front-end causes security to be activated for the Utilities. If Utility Security is installed, an SSF must be set up for the ADABAS nucleus ("NUC").

IX.5.4 Modify "STRPARM" Module

Modify the "STRPARM" module created in the installation of SECURITRE for ADABAS file security to have the utility parameters desired. For more information about the parameter statements used, refer to **Section IV - SECURITRE for ADABAS Utilities**. This "STRPARM" is used by the SECURITRE ADARUN front-end module used for ADABAS Utility Control. A sample "STRPARM" module is provided in STR.V331.SOURCE (STP00007).

There must be a "STRPARM" module for each ADARUN. The module name must conform to the following convention:

```
STP99999
```

In this case, 99999 is the DBID for that database. For example, the "STRPARM" module name for DBID=00007 is STP00007.

IX.5.5 Assemble STRPARM Module STRPARM

To assemble the "STRPARM" parameter module, the following JCL is provided as an example. However, it is recommended that installation personnel customize existing JCL to accomplish this step.

Note: When using the same JCL as in **Section X.4.3 - Assemble STRIOR Module**, uncomment the STRPARM assembly as shown here:

```
//*
//*STRIOR      EXEC   TSIASM,MEM=STRIOR
//*STNPNAT     EXEC   TSIASM,MEM=STNPNAT
//STP00251     EXEC   TSIASM,MEM=STP00251,LOAD='ADABAS.LOAD'
//*TSIEX4PR    EXEC   TSIASM,MEM=TSIEX4PR
//
```

Note: Retain the assembler listing in a safe place. TSI technical support personnel may ask for information from this listing to help resolve any problems that may arise.

IX.5.6 APF-Authorization

Because SECURITRE issues privileged SVCs to validate a user's ability to access a given file, the ADABAS and SECURITRE modules must reside in an APF-authorized library. This means that all of the loadlibs in the steplib for the ADABAS nucleus must be APF-authorized. If one loadlib is not APF-authorized, then all APF-authorization is lost. It is acceptable for modules to be loaded from a loadlib located in the "link list" if that library is APF-authorized. With this in mind, both the ADABAS loadlib and the SECURITRE loadlib must be APF-authorized. If only the ADABAS loadlib is APF-authorized, then the STRUEX1, STRUEX4, and STP99999 ("STRPARM") modules must be copied to the ADABAS loadlib. In addition, ADARUN must be re-linked with SETCODE AC(1).

The following JCL is located in STR.V331.SOURCE(JCLLKDA):

```
//*
//*     APF-AUTHORIZATION
//*
//LKED  EXEC    PGM=IEWL,PARM='MAP,LIST,NCAL'
//SYSLMOD       DD      DSN=ADABAS.LOAD,DISP=SHR
//ADALIB        DD      DSN=ADABAS.LOAD,DISP=SHR
//SYSUT1        DD      UNIT=SYSDA,SPACE=(CYL,(1,2))
//SYSPRINT      DD      SYSOUT=*
//SYSLIN        DD *
INCLUDE ADALIB(ADARUN)
SETCODE AC(1)
NAME    ADARUN(R)
//
```

IX.6. Install SECURITRE for NATURAL

Use the following checklist for the **optional** installation of SECURITRE for NATURAL under NATURAL. If SECURITRE for NATURAL is not desired, skip to the next section.

✓	STEP	FUNCTION	SECTION	REQUIRED
	N1	Apply any fix Zaps	IX.6.1	Y
	N2	Code STNPNAT	IX.6.2	Y
	N3	Assemble/Link STNPNAT	IX.6.3	Y
	N4	Modify the NATPARM module	IX.6.4	Y
	N5	Assemble/Link NATPARM module	IX.6.5	Y
	N6	Link-edit a NATURAL nucleus	IX.6.6	Y
	N7	Activate SECURITRE for NATURAL	IX.6.7	Y
	N8	Install SECURITRE for NATURAL Logon Security	IX.6.8	N

Repeat steps N2 through N8 for each NATURAL nucleus that will be secured by SECURITRE for NATURAL.

IX.6.1 Apply Any Fix Zaps

You may have received fix Zaps along with your release of SECURITRE. **All** relevant Zaps **must** be applied.

IX.6.2 Code STNPNAT

The STNPNAT module is provided to allow the Security Administrator to tailor SECURITRE for NATURAL to site-specific requirements. The STNPNAT module defines how SECURITRE will secure the NATURAL nucleus. A sample STNPNAT is provided with this release of SECURITRE in the source data set. Refer to ***Section III - SECURITRE for NATURAL*** in the ***SECURITRE Reference Manual*** for all the parameter values and defaults. Refer to **Section III - Setting Up SECURITRE For NATURAL** for examples of the use of these parameters.

IX.6.3 Assemble and Link STNPNAT

The same JCL used to assemble and link the STRIOR module can be used (refer to **Section IX.4.3 - Assemble STRIOR Module**). The default filename for the SECURITRE for NATURAL parameters is STNPNAT. There are no restrictions on the naming of the member containing the parameters. For details on changing this name refer to **Appendix A** of this manual.

IX.6.4 Modify the NATPARM Module

Modify the CSTATIC parameter to add the STNPRM entry. If a CSTATIC parameter does not exist, one must be added. An example of this parameter is shown below:

```
CSTATIC=(STNPRM)
```

If SECURITRE for NATURAL logon security is desired, STNLGN must also be added to the CSTATIC parameter. The CSTATIC parameter would then look like the following:

```
CSTATIC=(STNPRM,STNLGN)
```

Increase the value specified by USERBUF by the amount of space required by SECURITRE for NATURAL. The amount of space required is calculated as follows:

```
688 + (12 * PGMTBSZ)
```

The value calculated must be rounded up to the nearest 1K. For example, if 30 is used as the default value for PGMTBSZ, the increase to USERBUF would be calculated as follows:

```
688 + (12 * 30) = 1048
```

Since 1K is defined as 1024, the USERBUF parameter would be increased by 2. If a USERBUF parameter is not currently specified, NATURAL will use 2K as the default value. Therefore, 2K should be added to the default and the following line(s) would be added to the NATPARM module:

```
USERBUF=2
RDCSIZE=2
```

The PGMTBSZ is specified in the STNPARM parameter in STNPNAT that sets the number of programs to store in SECURITRE's internal tables per user. Refer to the Setting Up SECURITRE For NATURAL Section for more information on this parameter, and refer to the

NATURAL Installation & Operations Manual from Software AG for more information on specifying the CSTATIC and USERBUF parameters.

IX.6.5 Assemble and Link the NATPARM Module

The JCL used to assemble and link the NATPARM module when NATURAL was originally installed should be used.

Note: When assembling the NATPARM module with the USERBUF parameter specified with NATURAL 3.1.6 and above, the following message will appear in the listing:

```
USERBUF PARAMETER IS NO LONGER USED FOR THE DATA COLLECTOR.
PLEASE USE RCDSIZE INSTEAD.
```

This message may be ignored.

IX.6.6 Link-edit a NATURAL Nucleus

After the SECURITRE for NATURAL parameters and NATPARM module have been assembled, the NATURAL nucleus must be re-linked in one of the ways described below.

The installation personnel may make modifications to the existing NATURAL link JCL. A DDNAME of STRLIB must be added to point to STR.V331.LOAD.

When linking a NATURAL nucleus for TSO or Batch, insert all INCLUDE cards before 'INCLUDE ADALIB(ADAUSER)'. When linking a NATURAL nucleus for CICS, COM-PLETE, or TPF, insert all the INCLUDE cards before 'INCLUDE NATLIB(NATPM)'.

NATURAL 4.2 with a non-shared nucleus
The following INCLUDE cards are used for NATURAL 4.2 only.

```
INCLUDE STRLIB(TSIRDC42)
INCLUDE STRLIB(STN42A)
INCLUDE STRLIB(STN42B)
INCLUDE STRLIB(STN42D)
INCLUDE STRLIB(STN42E)
INCLUDE STRLIB(STN42GET)
INCLUDE STRLIB(STN42FRE)
INCLUDE STRLIB(STN42ZAP)
INCLUDE STRLIB(STN42ZAS)
INCLUDE STRLIB(STN42ZAL)
INCLUDE STRLIB(STN42LGN)  If logon security is desired
INCLUDE STRLIB(STN42RCH)
INCLUDE STRLIB(STN42PRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

NATURAL 4.2 with a shared nucleus

The following INCLUDE cards are used for the independent portion of a NATURAL 4.2 nucleus only:

```
INCLUDE STRLIB(TSIRDC42)
INCLUDE STRLIB(STN42A)
INCLUDE STRLIB(STN42B)
INCLUDE STRLIB(STN42D)
INCLUDE STRLIB(STN42E)
INCLUDE STRLIB(STN42FRE)
INCLUDE STRLIB(STN42ZAP)
INCLUDE STRLIB(STN42ZAS)
INCLUDE STRLIB(STN42ZAL)
INCLUDE STRLIB(STN42RCH)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

The following INCLUDE cards are used for the TP dependent portion of a NATURAL 4.2 nucleus only:

```
INCLUDE STRLIB(STN42A)
INCLUDE STRLIB(STN42GET)
INCLUDE STRLIB(STN42ZAL)
INCLUDE STRLIB(STN42LGN)  If logon security is desired
INCLUDE STRLIB(STN42RCH)
INCLUDE STRLIB(STN42PRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

NATURAL 4.1 with a non-shared nucleus

The following INCLUDE cards are used for NATURAL 4.1 only:

```
INCLUDE STRLIB(TSIRDC41)
INCLUDE STRLIB(STN4A)
INCLUDE STRLIB(STN4B)
INCLUDE STRLIB(STN4D)
INCLUDE STRLIB(STN4E)
INCLUDE STRLIB(STN4GET)
INCLUDE STRLIB(STN4FREE)
INCLUDE STRLIB(STN4ZAPP)
INCLUDE STRLIB(STN4ZAPS)
INCLUDE STRLIB(STN4ZAPL)
INCLUDE STRLIB(STN4LGN)   If logon security is desired
INCLUDE STRLIB(STN4RCHK)
INCLUDE STRLIB(STN4PRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

NATURAL 4.1 with a shared nucleus

The following INCLUDE cards are used for the independent portion of a NATURAL 4.1 nucleus only:

```
INCLUDE STRLIB(TSIRDC41)
INCLUDE STRLIB(STN4A)
INCLUDE STRLIB(STN4B)
INCLUDE STRLIB(STN4D)
INCLUDE STRLIB(STN4E)
INCLUDE STRLIB(STN4FREE)
INCLUDE STRLIB(STN4ZAPP)
INCLUDE STRLIB(STN4ZAPS)
INCLUDE STRLIB(STN4ZAPL)
INCLUDE STRLIB(STN4RCHK)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

The following INCLUDE cards are used for the TP dependent portion of a NATURAL 4.1 nucleus only:

```
INCLUDE STRLIB(STN4A)
INCLUDE STRLIB(STN4GET)
INCLUDE STRLIB(STN4ZAPL)
INCLUDE STRLIB(STN4LGN)   If logon security is desired
INCLUDE STRLIB(STN4RCHK)
INCLUDE STRLIB(STN4PRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

NATURAL 3.1.6 with a non-shared nucleus
The following INCLUDE cards are used for NATURAL 3.1.6:

```
INCLUDE STRLIB(TSIRDC31)
INCLUDE STRLIB(STNA)
INCLUDE STRLIB(STNB)
INCLUDE STRLIB(STND)
INCLUDE STRLIB(STNE)
INCLUDE STRLIB(STNGET)
INCLUDE STRLIB(STNFREE)
INCLUDE STRLIB(STNZAPP)
INCLUDE STRLIB(STNZAPS)
INCLUDE STRLIB(STNZAPL)
INCLUDE STRLIB(STNLGN)    If logon security is desired
INCLUDE STRLIB(STNRCHEK)
INCLUDE STRLIB(STNPRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

NATURAL 3.1.6 with a shared nucleus
The following INCLUDE cards are used for the independent portion of a NATURAL 3.1.6 nucleus:

```
INCLUDE STRLIB(TSIRDC31)
INCLUDE STRLIB(STNA)
INCLUDE STRLIB(STNB)
INCLUDE STRLIB(STND)
INCLUDE STRLIB(STNE)
INCLUDE STRLIB(STNFREE)
INCLUDE STRLIB(STNZAPP)
INCLUDE STRLIB(STNZAPS)
INCLUDE STRLIB(STNZAPL)
INCLUDE STRLIB(STNRCHEK)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

The following INCLUDE cards are used for the TP dependent portion of a NATURAL 3.1.6 nucleus:

```
INCLUDE STRLIB(STNA)
INCLUDE STRLIB(STNGET)
INCLUDE STRLIB(STNZAPL)
INCLUDE STRLIB(STNLGN)    If logon security is desired
INCLUDE STRLIB(STNRCHEK)
INCLUDE STRLIB(STNPRM)
INCLUDE STRLIB(STNPNAT)   If dynamic parameter loading is not used
```

IX.6.7 Activate SECURITRE for NATURAL

Note:	This section should be completed for all NATURAL environments.

Once a NATURAL nucleus has been linked with the SECURITRE for NATURAL components included, it is ready to be used, but the following items need to be considered:

- The database designated in the STNPARM SERVER parameter should be up and running with SECURITRE's ADABAS file security installed. Otherwise, all calls to the SSF will be rejected. Since the SSF rejects all calls, every command issued from NATURAL will result in a return code 200 and access to NATURAL will be denied.

- SSF rules should be set up for read access only. The one exception is where special logon privileges are requested. In this case an update rule should be defined. For more information, refer to Section III - Setting Up SECURITRE For NATURAL.

IX.6.8 Install SECURITRE for NATURAL Logon Security

SECURITRE for NATURAL logon security is an **optional** component. Before the following steps can be performed, the SECURITRE RTM must NATLOAD'ed. For more information refer to **Section IX.7.1 - Load SECURITRE RTM NATURAL Modules**.

1. Using SYSMAIN, rename the existing LOGON program to STRLOGON in the SYSLIB library of the FNAT file. **If an existing STRLOGON module exists, it should NOT be overwritten unless an upgrade to NATURAL occurred after logon security was installed.**

2. If 8-steplib support is desired, un-comment the statement 'STACK TOP COMMAND 'STRSTEP'' in the program STRLOGON in the STRLIB library. STOW the module.

Note:	This option can not be used if NATURAL security is installed.

3. Using SYSMAIN, rename STRLOGON to LOGON in the STRLIB library on the FUSER.

4. Copy the LOGON module from the STRLIB library on the FUSER to the SYSLIB library on the FNAT. Be sure to specify replace when copying the module.

5. Copy the STRLOGN map from STRLIB on the FUSER to the SYSLIB library on the FNAT.

6. Copy the STRSTEP program from the STRLIB library on the FUSER to the SYSTEM library on the FUSER.

7. Copy the STRCUST sub-program from the STRLIB library on the FUSER to the SYSTEM library on the FUSER. STRCUST is a source member that may be customized by the site. **This module should only be modified for special installation requirements or as directed by TSI.**

8. If the SECURITRE RTM is not installed, copy USR0050N, USR1025N, and USR2004N from the SYSEXT library on the FNAT to the SYSTEM library on the FUSER.

IX.7 RTM Installation

The RTM modules may be installed under NATURAL 3.1.6 or above.

> **Note:** It is recommended that the RTM be installed. Without the RTM, the database where SECURITRE is installed will need to be brought down and up if the "STRPARMS" or the SSF rules are modified to allow SECURITRE to access the new parms.

The following steps are necessary only if the SECURITRE RTM is desired:

R1 NATLOAD the RTM modules into NATURAL 3.1.6 or above. (Refer to section IX.7.1.)

R2 Execute STRUXCPY to copy the necessary USR* modules from the SYSEXT library to the SYSTEM library. (Refer to section IX.7.2.)

R3 Verify the NATLOAD and the STRUXCPY procedures. (Refer to section IX.7.3.)

R4 Modify the “STRPARM” module. (Refer to section IX.7.4.)

R5 Assemble and link the “STRPARM” module. (Refer to section IX.7.5.)

R6 Modify the NATPARM module. (Refer to section IX.7.6.)

R7 Assemble and link the new NATPARM module. (Refer to section IX.7.7.)

R8 Link-edit a NATURAL nucleus. (Refer to Section IX.7.8.)

R9 Define rules to SSF. (Refer to section IX.7.9.)

Use the following checklist for the **optional** installation of SECURITRE’s Real-Time Monitor (RTM) under NATURAL 3.1.6 and above:

✓	STEP	FUNCTION	SECTION	REQUIRED
	R1	NATLOAD the RTM modules	IX.7.1	Y
	R2	Execute STRUXCPY	IX.7.2	Y
	R3	Verify the installation	IX.7.3	Y
	R4	Modify the "STRPARM" module	IX.7.4	Y
	R5	Assemble/link the new "STRPARM" module	IX.7.5	Y
	R6	Modify the NATPARM module	IX.7.6	Y
	R7	Assemble/Link the NATPARM module	IX.7.7	Y
	R8	Link-edit a NATURAL nucleus	IX.7.8	Y
	R9	Define rules to SSF	IX.7.9	Y

IX.7.1 LOAD SECURITRE RTM NATURAL Modules

In order to install NATURAL, each installation typically has a "standard NATLOAD procedure" set up. Use this NATURAL NATLOAD procedure to place the SECURITRE RTM NATURAL modules into library STRLIB.

Sample NATLOAD related JCL (located in "STR.V331.SOURCE(JCLNLOAD)") and NATURAL commands to load the NATURAL modules for the RTM in a NATURAL environment follow:

```
//*     MEMBER(JCLNLOAD)
//*     INSTALL SECURITRE NATURAL MODULES
//*
//LOAD  EXEC    NATURAL
//CMWKF01       DD      DSN=STR.V331.RTMNTLD,VOL=SER=STR331,DISP=OLD,
//                      UNIT=TAPE,LABEL=(3,SL)
//CMSYNIN       DD *
LOGON STRLIB
NATLOAD ALL,*,FM,LIB,*
FIN
/*
//
```

The JCL above assumes a PROC "NATURAL" exists as the installation standard.

IX.7.2 Execute STRUXCPY to Copy the Necessary USR* Modules

This version of SECURITRE for NATURAL requires several NATURAL user-exit modules. To install these modules, logon to the STRLIB library and execute the program STRUXCPY. This program will copy USR0050N, USR1022N, USR1025N, USR1043N, and USR2004N from the SYSEXT library on the FNAT to the SYSTEM library on the FUSER.

IX.7.3 Verify the NATLOAD and the STRUXCPY Procedures

Logon to the STRLIB library and use ‘LIST * *’ to determine if the RTM modules are present. There should be 73 (MENU and all others starting with STR) objects including source for STRCUST.

Logon to the SYSTEM library and use ‘LIST * USR*’ to determine if the USR modules are present. The following modules should be listed:

```
USR0050N, USR1022N, USR1025N, USR1043N, and USR2004N
```

If other USR* modules are used by the site, they will also be listed.

IX.7.4 Modify the "STRPARM" Module

The RTM security is not functional until an STRRTM parameter and a RTMORDR parameter are defined within the STRDEF statement of the "STRPARM" module. These parameters are used to control access to the SECURITRE RTM. For more information on these parameters refer to **Section II.9 - Securing the RTM**.

> **Note:** The MODE for the RTM is always FAIL. RTM security requires READ access. When installing SECURITRE and attempting to use the RTM for the first time, DBAs or Security Administrators can "pre-approve" themselves by setting the STRRTM parameter to a high-level qualifier they are already allowed to access. Refer to **Section II.9 - Securing the RTM**.

IX.7.5 Assemble and Link the New "STRPARM" Module

After the "STRPARM" module is modified, it must be assembled and linked. The same JCL used to assemble and link the "STRPARM" module during the SECURITRE for ADABAS file security can be used. Refer to **Section IX.4.7 - Assemble STRPARM Module**.

IX.7.6 Modify the NATPARM Module

If the NTDB parameter is not currently defined in the NATPARM module, add it at this time. This parameter defines the ADABAS version for each of the databases that the RTM will access. The following is an example of what the NTDB parameter should look like:

```
NTDB ADAV5,(5,6,8,10)
NTDB ADAV6,(12,99,1000)
```

The example above specifies databases 5, 6, 8, and 10 as ADABAS version 5 databases and databases 12, 99, and 1000 as ADABAS version 6 databases.

For more information on the NTDB parameter, refer to the NATURAL Installation & Operations Manual from Software AG.

IX.7.7 Assemble and Link the new NATPARM Module

The JCL used to assemble and link the NATPARM module when NATURAL was originally installed should be used.

IX.7.8 Link-edit a NATURAL Nucleus

After the NATPARM module has been assembled, the NATURAL nucleus must be re-linked. The same JCL used in **Section IX.6.6 - Link-edit a NATURAL Nucleus** should be used.

IX.7.9 Define Rules to SSF

Define rules to allow access to the RTM based on the STRDEF STRRTM and RTMORDR parameters. Refer to **Section II.9 - Securing the RTM.**

IX.8 Initialization Problem Analysis

SECURITRE User-Exit-1 Problem Analysis:

SECURITRE User-Exit-1 will ABEND with a user ABEND code during the initialization processing if any abnormal conditions are encountered.

Depending upon the status of ADABAS (ESTAE active or dormant) at the time SECURITRE ABENDs, the SECURITRE user ABEND code will be printed in either hex or decimal. The following list of ABEND codes includes both the hex and decimal values of the SECURITRE user ABEND code, a description of the cause of each ABEND, and the action required for correcting it.

User ABEND Code: Decimal 300 (Hex 12C)
Cause: The expiration date for the trial has passed.
Action: Contact TSI.

User ABEND Code: Decimal 301 (Hex 12D)
Cause: SECURITRE could not allocate the RACROUTE work area for the SSF.
Action: Increase the region size by at least 512 bytes.

User ABEND Code: Decimal 302 (Hex 12E)
Cause: Sufficient storage space was not available for SECURITRE User-Exit-1 to issue its allocate requests.
Action: Obtain the size of these requests from the STRPARM assembly listing and increase the region size to accommodate them. For more information, refer to **Section X.2 - Storage Requirements** or decrease the amount of storage space SECURITRE requires by lowering the value of the USERS parameter.

User ABEND Code: Decimal 303 (Hex 12F)
Cause: SECURITRE User-Exit-1 to ADABAS could not load the STRPARM (STRP999) module for this database.
Action: Verify that the "STRPARM" module is in the steplib and is named according to the "STRPARM" naming conventions. For more information, refer to **Section IX.4.7 - Assemble STRPARM Module**.

User ABEND Code: Decimal 304 (Hex 130)
Cause: SECURITRE detected invalid file defaults (STRPARMs).
Action: Contact TSI.

User ABEND Code: Decimal 305 (Hex 131)
Cause: The "STRPARM" module did not assemble with a condition code of zero.
Action: Debug and re-assemble the module.

User ABEND Code: Decimal 306 (Hex 132)
Cause: SECURITRE is not APF-authorized.
Action: Ensure that SECURITRE is APF-authorized and try again. For more information, refer to **Section IX.4.8 – APF-Authorization**.

User ABEND Code: Decimal 307 (Hex 133)
Cause: SECURITRE could not determine the ADABAS version for the database.
Action: Contact TSI.

User ABEND Code: Decimal 308 (Hex 134)
Cause: SECURITRE detected an invalid value for the USERID parameter in the STRPARM module.
Action: Examine the assembler listing to verify that the value stated for the USERID parameter is valid. For more information about valid values, refer to **Section II.4.1 – Setting Up Database and File Security Parameters**.

User ABEND Code: Decimal 309 (Hex 135)
Cause: SECURITRE detected an error during a LOAD for a User-Exit.
Action: Examine the SECURITRE error message to determine which module SECURITRE cannot load.

User ABEND Code: Decimal 400 (Hex 190)
Cause: SECURITRE detected an internal error within the user table.
Action: Obtain a dump and contact TSI.

TSIUEX4 will ABEND during the initialization processing if any abnormal conditions are encountered. The following is a list of ABEND codes, a description of the cause of each ABEND and the action required for correcting it.

ABEND Code: U500
Cause: TSIUEX4 could not determine the ADABAS version installed.
Action: Contact TSI.

ABEND Code: U501
Cause: The TSIEX4PR parameter module was not link-edited with TSIEX4DR.
Action: Link-edit the TSIEX4PR parameter module with the TSIEX4DR dispatcher module and restart the database.

ABEND Code: U502
Cause: The module link-edited as the TSIEX4PR parameter module was not created with the TSIEX4P macro.
Action: Link Edit a valid TSIEX4PR parameter module with the TSIEX4DR dispatcher module and restart the database. The link-edit order must be TSIEX4DR, followed by TSIEX4PR.

ABEND Code: U503
Cause: The module link-edited as the TSIEX4PR parameter module did not assemble with condition code 0.
Action: Correct any assemble errors and re-assemble the TSIEX4PR module. Re-link-edit a valid TSIEX4PR parameter module with the TSIEX4DR dispatcher module and restart the database.

ABEND Code: S806
Cause: Either the TSIUEX4 module for this database could not be found by ADABAS, or the TSIUEX4 could not load one of the User-Exit-4 programs specified in the TSIEX4PR module.
Action: Examine the system error message to determine which module could not be loaded and the reason it cannot load this module.

If problems are encountered during SECURITRE for NATURAL startup, NATURAL exits with message 9987 and one of the following response codes:

100: NSI Security failure

101: Error during buffer processing or STNGET not found

102: Dynamic Parameter load requested and parameters not found

103: Dynamic Parameter load NOT requested; parameters not linked

104: STNBUF not found in CSTATIC list

105: Incompatible NATURAL version detected

106: ADALNK is not found

SECTION X

OPERATIONS CONSIDERATIONS

X.1 SECURITRE Execution Requirements

The ADABAS parameters required are:

```
UEX1=STRUEX1
UEX4=STRUEX4      (or TSIUEX4)
LOGGING=YES
```

SECURITRE requires NATURAL at Version 3.1.6 or above for running the RTM and/or SECURITRE for NATURAL.

X.2 Storage Requirements

The Region Size for the ADABAS nucleus must be adequate for SECURITRE to operate. The SECURITRE region requirements can be broken down into the following categories:

- The SECURITRE User-Exit-1 requires approximately 34K for its "base" code.
- The STRPARM module will be loaded by SECURITRE User-Exit-1. This module requires approximately 20K, plus 10 bytes for each field being checked for field level security.
- The SECURITRE User-Exit-1 will issue allocate requests for storage. The amount of storage required for these allocate requests can be found in the assembly listing of the "STRPARM" module.

  In general, the amount of allocate storage required is 512 bytes + (USERS * 68) bytes for the USER table, plus (DSNPOOL*52) bytes for the DSN table, plus (USRPOOL*275) bytes for the USER/DSN relationship table, plus USERS*((FLSPOOL/10 * 25) + (FLSPOOL * 10)) bytes for the FLSCID table.
- If another User-Exit-1 is to be executed in addition to SECURITRE or if any of the SECURITRE user-exits have been selected, there must be enough region size available for these modules to be loaded.
- The SECURITRE User-Exit-4 requires approximately 4K.
- The SECURITRE ADARUN front-end for ADABAS Utility Control requires 4K. This module loads the STRPARM module as described above.
- If the TSIUEX4 module is used, it requires 4K. Enough region size must be available for the modules it loads.

SECURITRE User-Exit-Bs require small amounts of storage. These are linked with the appropriate ADABAS Link Routines.

X.3 Problem Solving Checklist

For more information about problems that occur during the initialization of SECURITRE User-Exit-1 and Utility Security, refer to the Initialization Problem Analysis section of this manual. For more information about problems involving User-Exit-4 co-existence or the TSIUEX4 "dispatcher" program, refer to **Section VIII.2 - User-Exit-4 Co-existence**. If these sections do not contain the solution to the problem being experienced, the following checklist may help to diagnose or solve the problem.

- Has new Software AG software, IBM software, Security software, or any system maintenance recently been added to the system? If so, what?

  New software or system maintenance may affect SECURITRE and its ability to function in a user's environment. Try running SECURITRE with the new software removed, if possible. If the problem goes away, the new software could be the cause. Call TSI for more information.

- Are all TSI published Zaps applied? Are they applied correctly?

  The failure to apply all published Zaps or the incorrect application of a published Zap can have a severe negative impact on the operation of SECURITRE. Verify all Zaps. If the problem persists, contact TSI. At that time, verification that all published Zaps have been received can be made.

- If an ABEND occurred, did any particular event seem to cause the ABEND?

  Make a note of the event causing the ABEND, and then contact TSI.

- Can the problem be re-created? Can it be done easily?

  Make a note of how the problem can be re-created and contact TSI.

- If an ABEND occurred, does the ABEND occur across all databases? If not, what are the differences between the databases? Do all databases have SECURITRE user-exits operational? Is the same User-Exit-1 being used across all databases?

  It is important that the SECURITRE versions are consistent across databases. Mixed versions can be unpredictable.

- Is another User-Exit-1 being used? Has this other user-exit been checked out by itself or with ADABAS?

  If the other user-exit works on its own, contact TSI.

- Is this the first time this problem has occurred under this version of SECURITRE?

  If this version of SECURITRE has been running consistently without problems and suddenly ABENDs, something may have changed. Installation of new software, application of maintenance, reorganizing load libraries, and other such changes could contribute to the problem and should be considered when attempting to solve the problem that is occurring.

- Is this the first attempt to run a new version of SECURITRE? If yes, did the problem occur under an older version?

  If the new version of SECURITRE is incorrectly installed, an ABEND could result. Verify the installation before calling TSI.

  If the problem did not occur under an older version of SECURITRE, contact TSI to determine why the problem might have occurred.

- Is it a problem with ADABAS Utilities execution?

  Refer to **Section IV – SECURITRE for ADABAS Utilities** and ***Section IV – SECURITRE for ADABAS Utilities*** in the ***SECURITRE Reference Manual***.

- Does the problem still occur after SECURITRE is removed?
  If the problem persists after SECURITRE is removed, the cause of the problem is not SECURITRE. Other areas should be researched.

Before calling TSI, be sure to have the ADAM99 information from the ADABAS SYSLOG available, along with the answers to the questions above. This information will greatly improve the speed of the resolution of the problem.

X.4 WTO Messages

SECURITRE issues various messages during the start-up process and during the ADABAS nucleus session.

During SECURITRE startup on database 007 the following messages will appear:

```
STRUX1 00007 STRV331    SECURITRE STARTUP IS IN PROGRESS
STRUX1 00007 STRV331    SECURITRE LOAD POINT AT   : 0009D190
STRUX1 00007 STRV331    STRP007 MODULE ASSEMBLED  : 06/01/06
STRUX1 00007 STRV331    SECURITRE IS ACTIVE
STRUX1 00007 STRV331    FOR DATABASE              : TSI-ADA-DEVL
```

During SECURITRE start-up the following messages may appear if user-exits have been specified:

```
STRUX1 00007 STRV331    SECURITRE LOADED PROGRAM  : TRMUEX1
STRUX1 00007 STRV331    SECURITRE LOADED PROGRAM  : NOUSER
```

If purge intervals are specified, the following messages may appear:

```
STRUX1 00007 STRV331    TABLE INTERVAL PURGE COMPLETED
STRUX1 00007 STRV331    NUMBER OF USERS REMOVED   : 00009
```

X.5 STRMSG Messages

If the STRMSG DDNAME has been specified in the startup JCL for the ADABAS nucleus, SECURITRE will print assorted diagnostic information to this DDNAME. This information will include the date of assembly, time of assembly, and the Zap status of each component of SECURITRE.

X.5.1 Zap Status

The STRMSG dataset will contain assorted diagnostic information. This information will include the date of assembly, time of assembly, and Zap status of each component of SECURITRE. The following listing illustrates this information:

```
                T R E E H O U S E   S O F T W A R E
                S E C U R I T R E   M E S S A G E S


MODULE=STRMAIN    ASSEMBLE DATE : 06/01/06   ASSEMBLE TIME : 17.28

ZAPS APPLIED      NONE

MODULE=STRINIT    ASSEMBLE DATE : 06/01/06   ASSEMBLE TIME : 17.26

ZAPS APPLIED      01

.
.
.

MODULE STRTRAC    ASSEMBLE DATE : 06/01/06   ASSEMBLE TIME : 17.29

ZAPS APPLIED      NONE

MODULE STRERR     ASSEMBLE DATE : 06/01/06   ASSEMBLE TIME : 17.26

ZAPS APPLIED   01,03,07
-----------------------------------------------------
```

The information in the STRMSG dataset will be useful in the event that a problem occurs with SECURITRE because it will enable the TSI support staff to determine the cause of the problem more rapidly. It will also enable the user to easily determine the specific maintenance level of the software.

X.5.2 Trace Messages

If the SECURITRE Trace Facility has been activated, SECURITRE will write the diagnostic trace messages to the STRMSG dataset. The number and type of TRACE messages will be controlled by the method used to turn the trace on. The following listing is an example of the messages produced by the trace facility:

```
                T R E E H O U S E   S O F T W A R E
                S E C U R I T R E   M E S S A G E S


DATE AND TIME COMMAND RECEIVED IN STRUEX1:99 - 215  16:38:35
TRACE PT:001      USERID:           GROUP:   FILE:243 CMD:L3
-------------------------------------------------------------
TRACE PT:002      USERID:           GROUP:    FILE:243 CMD:L3
DSNORDR:GJOB,FILE,FLD                         OPTIONS:WAAA
-------------------------------------------------------------
TRACE PT:003      USERID:STRV331      GROUP:   FILE:243 CMD:L3
-------------------------------------------------------------
TRACE PT:008      USERID:STRV331      GROUP:   FILE:243 CMD:L3
DSN:ADABAS.PROD.PAYROLL
-------------------------------------------------------------
RACROUTE REQUEST=AUTH,ATTR=READ,CLASS=DATASET,USERID=STRV331
         ENTITY=ADABAS.PROD.PAYROLL
-------------------------------------------------------------
RACROUTE REQUEST=AUTH,ATTR=READ,CLASS=DATASET,USERID=STRV331
         ENTITY=ADABAS.PROD.PAYROLL.PAYRATE
-------------------------------------------------------------
TRACE PT:004      USERID:STRV331      GROUP:    FILE:243 CMD:L3
ACTION:0 REQUEST ALLOWED
-------------------------------------------------------------
.
.
.
```

Activating the trace has the potential to create a substantial increase in overhead. Therefore, it is suggested that the trace facility should only be turned on while testing or debugging is taking place.

The trace output consists of informational messages written at various points during the execution of the SECURITRE User-Exit-1 to ADABAS. With the exception of the SSF interface trace, the trace points are numbered 1 to 8 and print out the trace point numbers, User-ID, group, file number, and ADABAS command. Trace points 1 and 2 will display a blank User-ID/group because these trace points are reached before this information is obtained.

The following chart lists where these trace points are taken and the additional information that is printed with them:

Trace Point	**Point of Execution**	**Additional Information Printed**
1	Entry to User-Exit-1	Date and Time command was received in STRUEX1
2	After file parameters are obtained	DSNORDR Options: File Mode D=DORMANT W=WARN F=FAIL Log Violations A=ALL F=FIRST NOID Update R=REJECT A=ACCEPT NOID Read R=REJECT A=ACCEPT
3	After User-ID is obtained	none
4	Leaving User-Exit-1	Action: return code from STREX1 and explanation
5	Beginning of user table reorganization	"REORG BEGIN"
6	End of user table reorganization	"REORG END"
7	After User-ID is looked up in the user table	"USER FOUND IN TABLE" or "USER ADDED TO TABLE" or "*** TABLE ERROR ***"
8	After pseudo DSN is built	pseudo DSN

If a trace on the SSF interface is requested, the following information will be printed:

- request type (auth, create, delete)
- attr (read, write)
- class
- User-ID
- entity (pseudo DSN)

The trace may be activated in different ways. First, by setting the STRDEF parameter to ON, all trace points will be activated for all files, all commands, and all users. Setting the STRDEF TRACE=OFF and setting the STRFNR TRACE=ON for individual files will result in all trace points being activated for those files.

To reduce the amount of trace information generated, the RTM TRAC function may be used to specify that only individual trace points should be printed. For instance, if information on pseudo-DSNs and STRUEX1 results is desired, the TRAC function would be used to indicate that trace points 4 and 8 be invoked. At that point, it is necessary to use the RTM PARM function to turn the TRACE parameter ON for the files being used. The trace may later be turned off by setting it to OFF using the PARM function for individual files or by turning the STRDEF and STRFNR TRACE parameters off and reassembling and reloading the parameters.

The amount of trace information generated may be further reduced, using the TRAC function, by indicating that the trace be printed only for a specific User-ID and for specific ADABAS commands. There will still be some machine overhead, but the amount of output will be cut down.

X.6 SECURITRE Questions and Answers

TSI salespeople, technicians, and affiliates are in constant communication with the user community. Often, users ask questions that we want to share with others. The following questions come from correspondence between TSI and potential SECURITRE customers.

Q. Does NATURAL Session Initialization Security provide a way to force a user that specifies a given FUSER file to use a corresponding FDIC file?

A. SECURITRE currently provides the capability to force a user to specify corresponding FUSER and FDIC files. Consider this example:

A site uses three NATURAL modules, the first is for the TEST environment, the second is for the STAGE environment, and the third is for the PROD environment. The Security Administrator codes the following SECURITRE for NATURAL parameters for the TEST environment:

Col. 72

```
STNPARM PREFIX='NATURAL',QUAL='TEST',                                  x
        DELIM='.',NSIORDR=(FILE)
STNFILE TST,DBID=123,FNR=200      (TEST FDIC)
STNFILE TST,DBID=123,FNR=201      (TEST FNAT)
STNFILE TST,DBID=123,FNR=202      (TEST FUSER)
```

To grant access to the TEST environment, the Security Administrator grants a user access to the following pseudo dataset name in the SSF:

```
NATURAL.TEST.TST
```

(continued on next page)

(continued from previous page)

If the user attempts to initiate a session from the TEST NATURAL module using the TEST FNAT, STAGE FDIC, and PROD FUSER, SECURITRE verifies access to these pseudo dataset names:

```
NATURAL.TEST.TST
NATURAL.TEST.STG
NATURAL.TEST.PRD
```

Because the Security Administrator did not grant the user access to all three of these dataset names, the session will not begin. In order to initiate a NATURAL session, the user must specify an authorized set of FNAT, FUSER, and FDIC files.

Q. Must a site specify STNFILE parameters for ALL files, including data files?

A. No. If STNFILE parameters are not specified, SECURITRE generates default pseudo dataset names based on the DBID/FNR combination. STNFILE statements provide "alias" names for files to simplify security administration. For example, rather than granting access to the cryptic pseudo dataset name "D213F123", the Security Administrator can grant access to the more meaningful database name, "TEST" and the "PAYROLL" file. "TEST.PAYROLL" is the resulting pseudo dataset name.

Q. What library name is placed in the pseudo dataset name when a user executes a program from a STEPLIBed library?

A. Currently, SECURITRE inserts the name of the library that the user is logged on to. We can provide a Zap to insert the name of the library where the program resides if necessary.

Q. If USRMODE=OFF is specified for a library in the STNLIB parameters and a user logs on to that library (Lib-A), then logs on to another library (Lib-B), and runs a program that sets NC=OFF, which is equivalent to USRMODE=ON, will the user be able to log back on to Lib-A and execute NATURAL commands?

A. No. When the user logs back on to Lib-A, SECURITRE sets NC=ON, which is equivalent to USRMODE=OFF.

Q. In programs with highly sensitive data, we would like to re-verify the user's password before the user is allowed to examine or modify data. Can SECURITRE help with this?

A. Yes. Before data is displayed or modified in your programs, you can request the password from the user and code a call to STRNAT. The pseudo dataset name might look like this:

```
NATURAL.PASSWORD.pswd
```

In the example above "pswd" is the password entered by the user. If the SSF grants the user access to this dataset, the program can assume that this is the correct password for this user and allow the user to display or to modify the data. This provides the advantages of ensuring that the password given by the user is correct, and that the user giving the password is actually authorized to use that password.

(continued on next page)

(continued from previous page)

Q. We use a performance monitor other than TRIM. Does SECURITRE co-exist with it?

A. Several TSI customers run SECURITRE with other performance monitors, and none of them have reported any co-existence problems. We do not currently know of any co-existence problems between other performance monitors and any TSI product.

Q. If I add a DDM to the FDIC and forget to grant a user access to the DDM, must CICS be brought down to give that user access to the DDM?

A. No. Grant the user access to the DDM through the System Security Facility and purge the user's entry from the SECURITRE internal table using the Real-Time Monitor. The user will have access to the DDM on the user's next attempt.

Q. I notice that SECURITRE has a very impressive list of features, including several levels of NATURAL access control, such as Program and Library Level security. If I install SECURITRE, do I have to use all of these features?

A. No. SECURITRE does not determine what will be secured. You do. The various features may be used separately. For example, you might only use Session Initialization and Program Level security on the NATURAL side and choose not to use DDM Level security. If you later decide to discontinue use of Program Level security or to begin using DDM Level security, you may do so. SECURITRE will grow and change with your site's security needs.

Q. Why does logging occur with CMDLOG=OFF specified in the STRPARMS?

A. User-Exit-B is not installed in all ADABAS link routines being used or User-Exit-4 is not properly installed. For more information refer to the ADABAS Link Routine User-Exits B and A section of this manual and activate the SECURITRE for ADABAS section of this manual.

Q. Why do I receive a NAT0920 error when I am using the SECURITRE RTM?

A. The STNPRM module has not been included in the CSTATIC parameter of the NATPARM module. For more information refer to **Section IX.6.4 - Modify the NATPARM Module**.

Q. Why do I receive a NAT0082 error when I am using the SECUIRTRE RTM?

A. The most common reason for this error is that the STRUXCPY program did not successfully execute during the RTM installation phase. For more information refer to **Section IX.7 - RTM Installation**.

Q. Why do I receive a NAT0082 error when logging onto a library when 8-steplib support is installed?

A. One of the following modules were not copied to the appropriate library: STRSTEP, STRCUST, STRLOGN, USR1025N, or USR200N. For more information refer to **Section IX.6 - Install SECURITRE for NATURAL**.

Q. Why does SECURITRE allow commands to be processed when MODE=FAIL is specified in the "STRPARMS?"

A. For more information refer to the MODE parameter in the chart in ***Section II - SECURITRE for ADABAS Nucleus*** in the ***reference manual***.

This page intentionally left blank.

APPENDIX A

Most sites should link the SECURITRE for NATURAL parameters (STNPNAT) to the NATURAL nucleus. If dynamic loading of these parameters when NATURAL is invoked is required for Batch and TSO follow the instructions listed below in reference to dynamic parameter loading.

Statically Linked SECURITRE for NATURAL Parameters

The default name for the SECURITRE for NATURAL parameters is STNPNAT. For sites with many NATURAL nuclei that require many different sets of SECURITRE for NATURAL parameters, the name STNPNAT may be changed to any name. When linking the NATURAL nucleus change the include:

```
INCLUDE STRLIB (STNPNAT)
```

to:

```
INCLUDE STRLIB (xxxxxxx) where XXXXXXX is the new name of the SECURITRE for
NATURAL parameters desired for this NATURAL nucleus
```

Dynamic Parameter Load for SECURITRE for NATURAL Parameters

Most sites should link the SECURITRE for NATURAL parameters (STNPNAT) to the NATURAL nucleus. If dynamic loading of these parameters when NATURAL invoked is required for BATCH and TSO, the following Zap should be applied to the STNA module for NATURAL 3.1.6 and the STN4A module for NATURAL 4.1.

For NATURAL 4.1 only:

```
NAME       STN4A   STNA
VER 0131   00
REP 0131   FF
```

For NATURAL 3.1.6:

```
NAME       STNA    STNA
VER 0131   00
REP 0131   FF
```

The default is to have the parameters statically linked. The dynamic parameter loading option should only be used during SECURITRE for NATURAL testing.

To have the parameters dynamically loaded, every "INCLUDE STRLIB (STNPNAT)" statement must be removed from the NATURAL link-edit.

The default filename for the SECURITRE for NATURAL parameters is STNPNAT. If this module name is changed, the following Zap must be applied.

For NATURAL 4.1 installations:

```
NAME       STN4A   STNA
VER 0138   E2E3,D7D5,D7C1,E340    STNPNAT
REP 0138   xxxx,xxxx,xxxx,xxxx    New module name
```

For NATURAL 3.1.6 installations:

```
NAME        STNA    STNA
VER 0138    E2E3,D7D5,D7C1,E340    STNPNAT
REP 0138    xxxx,xxxx,xxxx,xxxx    New module name
```

The REP value in the Zap above represents the hexadecimal notation for the new name of the module. For example, if the SECURITRE for NATURAL parameter module will be called STNPTST and installed with NATURAL 3.1.6, the Zap would be applied as follows:

```
NAME        STNA    STNA
VER 0138    E2E3,D7D5,D7C1,E340    STNPNAT
REP 0138    E2E3,D7D5,E2E3,E240    STNPTST
```

If the name is less than eight characters long (STNPTST is seven), the remaining bytes must be Zapped to spaces (hex 40).

If the site will use several different parameter modules for different NATURAL nuclei, separate STNA (or STN4A) modules should be kept in different load libraries. Each module will contain a different name at the Zap location above.

APPENDIX B

SECURITRE Co-exists with APAS

If SECURITRE is to co-exist with APAS, skip **Section IX.4.9 - ADABAS Link Routine User-Exit B**. STRUEXB, STRUEXBB, STRUEXB2, STRUEXB3, and STRUEXB5 should not be used. Instead, the APAS User-Exit-B should be used. Use the following steps to perform this install:

Installation of ADALNC, ADALNK, ADALCO, LNKOLM & LNKOLSC:

Only the APAS User-Exit-B's are used with ADALINK. In the source for DBGLNC5, the variable &ACCOUNT should be set to 'CICS.' In the source for DBGLNK5, the variable &ACCOUNT should be set to 'RACF.'

STRPARM assignments:

The STRDEF parameter USERID should be set to ALT-2. Do not assign anything to USERID2.

Installation of User-Exit-4:

The SECURITRE User-Exit-4 Dispatcher should be installed according to the instructions in **Section VIII.2 - User-Exit-4 Co-Existence**. The TSIEX4P parameters should include UEX4NI-APASUEX4.

This page intentionally left blank.

GLOSSARY OF TERMS

ACEE — An Accessor Environment Element is an IBM control block that describes the current user, including User-ID, current connect group, user attributes, and group authorities. It is constructed during user identification and verification. The ACEE is where SECURITRE obtains the User-ID.

ADALINK — ADABAS Link Routine. When an application calls "ADABAS," it actually calls an ADALINK module, which in turn "calls" the ADABAS SVC to communicate with the ADABAS address space. ADALNK is the name of the Batch/TSO ADALINK module. ADALNC is for CICS prior to 3.2, LNKOLM and LNKOLSC for CICS 3.2 and above, ADALNI is for IMS, and TLOPADAB is for COMPLETE.

ADARUN — ADABAS module used for all ADABAS Utility runs. In SECURITRE, this module is renamed RUNADA and replaced with an ADARUN module provided with SECURITRE to enforce ADABAS Utility Security.

CA-ACF2 — (ACF2) A System Security Facility sold by Computer Associates.

CA-TOP SECRET — (TOP SECRET) A System Security Facility sold by Computer Associates.

DDM — Data Definition Module. This is a collection of field descriptions, which may include all the fields in a file, a partial set of fields, or a "view" of a file, sometimes referred to as a "userview." NATURAL programs reference DDMs to access ADABAS files (or views) and individual fields. In order for a NATURAL program to be compiled, all files/views/fields must be defined in DDMs.

DDM Security — A SECURITRE for NATURAL feature for securing compile-time access to DDMs.

DELIM — A DELIMiter. For example, in the dataset name ADABAS.PROD.PAYROLL, the period (".") is the delimiter.

DORMANT — *See MODE.*

DSN — Pseudo dataset name or the entity rule passed to the System Security Facility for a security check.

FAIL — *See MODE.*

FDIC — The NATURAL System File (on ADABAS) containing DICtionary (PREDICT) data.

FNAT — The NATURAL System File on ADABAS primarily containing the NATURAL product.

FUSER — The NATURAL System File on ADABAS primarily containing the USER-written NATURAL code.

Logon Security	The SECURITRE for NATURAL feature for Logon (Library) Security.
MODE	The level of security protection desired. DORMANT mode allows for phasing in security control over databases, files, applications, etc. This mode does not hinder access. WARN mode is the next level, producing warning messages in the SSF log, but not hindering access. FAIL mode is the most stringent level, producing warning messages and preventing access.
NATURAL Security System	The Software AG product providing some control over the NATURAL environment.
NODE	The SMFID of the CPU from which an ADABAS call originates.
NOIDRED	This means NO User-ID is found for a REaD command to the database.
NOIDUPD	This means NO User-ID is found for an UPDate command to the database.
NSI Security	*See Session Initialization Security.*
NSS	Software AG product NATURAL SECURITY System.
PREFIX	The first part of the DSN to pass to the SSF, such as ADABAS in ADABAS.PROD.PAYROLL.
Program Security	The SECURITRE for NATURAL feature that controls the ability to READ, SAVE, CATALOG, or EXECUTE a NATURAL program.
Program Write Security	The SECURITRE for NATURAL feature that controls the ability for a specific user to write any NATURAL object while logged on to a library.
QUALIFY	The second part of the DSN to pass to the SSF, such as PROD in ADABAS.PROD.PAYROLL.
RACF	A System Security Facility sold by IBM.
RACFID	*See USERID.*
RACHECK	The call to the System Security Facility to determine a user's authorization to access a specific DSN.
RTM	SECURITRE Real-time Monitor.
RUN Security	The SECURITRE for NATURAL feature for securing the RUNning of NATURAL programs.
RUNADA	*See ADARUN.*
SAF	Security Authorization Facility protocol, a standard protocol for interfacing to RACF, ACF2, and TOP SECRET.

Security Administrator The person or staff responsible for securing the site's computer resources.

Session Initialization The SECURITRE for NATURAL feature that **Security** controls whether a user is allowed to initiate a session on a particular NATURAL nucleus.

STEPLIB The origin of the NATURAL module. The steplib is the current FUSER/Library, then SYSTEM, then up to 8 others.

STNPARM SECURITRE PDS object member containing assembled parameters for SECURITRE for NATURAL.

STRASM A callable routine (from 3GL languages) to provide miscellaneous application level security.

STRDEF SECURITRE parameter statement that defines DEFault settings used by SECURITRE for ADABAS and Utility security.

STREX1-4 User-Exits to SECURITRE, not SECURITRE exits to ADABAS.

STRFNR SECURITRE parameter statement for individual ADABAS files (FNRs), which override the settings that are established by STRDEF.

STRIOR The SECURITRE module specific to the SSF and operating system in use.

STRMSG A dataset that will contain SECURITRE messages of violations and trace information.

STRNAT A callable routine from NATURAL to provide miscellaneous application level security.

STRPARM SECURITRE PDS object member containing assembled parameters for SECURITRE for ADABAS.

System Security Facility (SSF) Any system-wide security facility that provides for a single rule base, centralized security, and control over the site's resources. Currently, SECURITRE supports these SSFs: RACF, CA-ACF2, and CA-TOP SECRET.

TRACE SECURITRE Trace Facility.

USERID, User-Id Unique ID for a user obtained from the ACEE.

USERINFO Information pertaining to the individual user, created in the SECURITRE User-Exit-B to the ADALINK routines, passed in a user-buffer to ADABAS, and referenced in SECURITRE User-Exit-1 and User-Exit-4 to ADABAS.

WARN *See MODE.*

WTO Write To Operator. WTO messages are important SECURITRE messages written to the Operator's Console.

This page intentionally left blank.

SECURITRE Administrator Guide Index

P

Q

R

S

T

U

This page intentionally left blank.