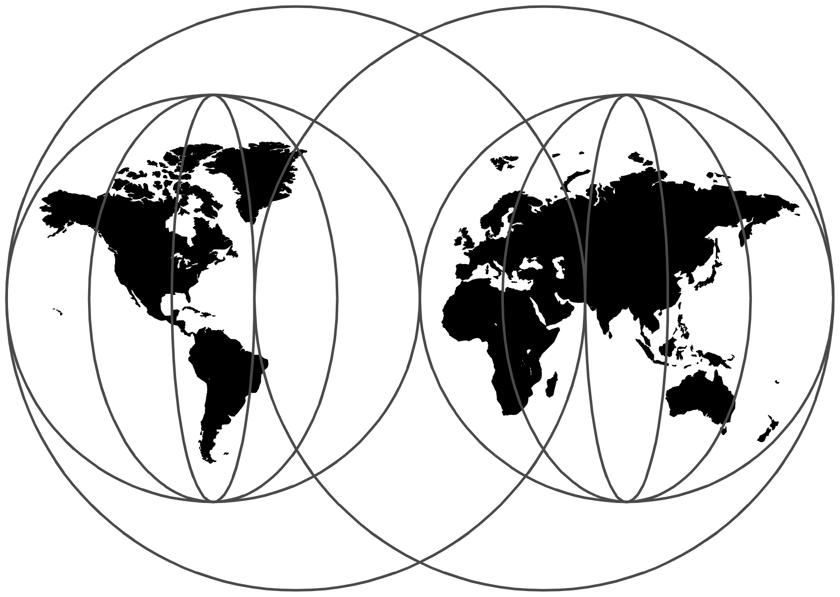
# Securing Applications with Tivoli Security Management Lockdown Modules

*Richard Hawes, Edson Manoel, Holger Wieprecht, Joseph Walczak, Thomas Sant'Ana*



**International Technical Support Organization**

http://www.redbooks.ibm.com

SG24-5140-00

SG24-5140-00

International Technical Support Organization

# Securing Applications with Tivoli Security Management Lockdown Modules

June 1999

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 119.

**First Edition (June 1999)**

This edition applies to Version 3.6 of Tivoli Security Management.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. OSJB  Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**© Copyright International Business Machines Corporation 1999.   All rights reserved.**
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

© Copyright IBM Corp. 1999

# Figures

© Copyright IBM Corp. 1999

# Tables

© Copyright IBM Corp. 1999

# Preface

A Tivoli Security Management solution can greatly enhance the role of a security administrator. Role-based security and cross-platform management ensure accurate, consistent, and efficient adherence to security policy.

However, the specification of access permissions for various resource types across multiple platforms is a complex task. Tivoli Security Management makes this task much simpler, but a sophisticated product such as Tivoli Security Management requires careful planning and testing. Also, having the capability to implement this protection across a wide range of resource types means identifying what needs to be protected and how best to implement that protection.

This redbook explains the concept of a *Lockdown Module* and how to go about determining how to protect applications. A lockdown module allows you to define access controls on resources and set up Tivoli security roles and groups and then convert that definition into a script. When run, this script will create all those definitions in a Tivoli Security Profile. This book demonstrates the use of lockdown modules for the distribution of ready-made security profiles for operating system platforms, applications, and even the Tivoli Framework itself.

The Lockdown Definition File (LDF) import utility and script implementation will be explored in more depth by Tivoli in the future, and we can expect to see more modules as this idea develops.

Chapter 1 contains an introduction to the topic of locking down applications and the use of lockdown modules. The modules explored in this redbook are included on the accompanying CD.

## The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

**Richard Hawes** is a Senior Tivoli Security Specialist at the International Technical Support Organization (ITSO), Austin Center. He writes extensively on Tivoli Framework and Security issues and is a Tivoli Certified Enterprise Consultant. Before joining the ITSO in early 1997, Richard worked for a number of years as a trouble-shooting consultant for an IBM European technical support team based in the UK.

**Edson Manoel** is a Tivoli Project Leader in the International Technical Support Organization (ITSO), Austin Center. He has many years of experience in network products especially in distributed technologies and networking protocols. Prior to joining the ITSO, Edson designed and implemented systems and network management solutions with IBM Global Services, Brazil.

**Holger Wieprecht** is a Systems Management Specialist based in Germany. He has several years of experience in the computer networks field. Holger has worked at IBM Global Services for four years. His areas of expertise include AIX, Concepts and Implementation of Secure Internet Gateways and Servers as well as Tivoli Systems Management.

**Joseph Walczak** is a Systems Engineer with Delta Technology/Delta Air Lines. He has 10 years of computer security experience. His areas of expertise include client/server security with both Windows NT and Unix as well as security architecture/infrastructure and Tivoli Systems Management. He has written several data communications security documents while in the U.S. Navy as a Cryptologic Technician and then as a government contractor.

**Thomas Sant'Ana** is an IT Specialist for Engesoftware Consultoria de Sistemas in Brazil. He has several years of experience in Tivoli deployment and application development. He holds a degree in Bachelor of Computer Science from Universidade de Brasilia. His areas of expertise include system security, database administration, and object-oriented application design and implementation.

Thanks to the following people for their invaluable contributions to this project:

Craig Sullivan
Mike Krajicek
Patrick McDowell
Stefan McKenzie
**Tivoli Systems**

Jochen Markus Weiss
**IBM Germany**

## Comments Welcome

**Congratulations or criticism, your comments are important to us!**

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 137 to the fax number shown on the form.
- Use the online evaluation form found at: `http://www.redbooks.ibm.com`
- Send us a note at the following address:

  `redbook@us.ibm.com`

# Chapter 1. Introduction

In order to provide maximum flexibility and improvements in administrator reliability and efficiency, Tivoli Security Management (TSM) abstracts away from the security model of individual platforms. We still manage the platform-specific security data, but we achieve cross-platform security modeling through *role-based* security. The benefits of this abstraction are significant, but it makes the implementation of the security model more difficult. An ideal TSM implementation design would include the identification of all job tasks in an organization as well as the IT resource access requirements to fulfill each of those tasks. This data would form the basis of TSM roles and resource data. All personnel would also be defined in TSM groups. In a large organization, this can take some time.

Through this redbook and its accompanying CD, we will introduce you to a faster method of improving the security of applications running under various operating system platforms. Most major applications have some form of security model within them, usually, to manage the authentication of users trying to access data. However, the data files and binaries of most applications are still subject to damage, whether intentional or not. This is especially true in the UNIX environment where, by default, there is nothing to prevent a root user from really messing up an application.

Tivoli Security Management (TSM) was the first security maintenance product designed from the ground up to manage access control in a consistent fashion in a distributed environment using a role-based security model. With role-based security, we determine what resources people have access to based upon the job tasks - or roles - that they need to perform. In TSM, these resources may be of many different types such as files, printers, programs or TCP services. They may also reside on different platform types such as Windows NT, various flavors of UNIX and on OS/390 systems protected by the OS/390 Security Server Resource Access Control Facility (RACF). Once the resources that are required to complete a task have been identified, they can all be listed in a role and the relevant access rights given. Tivoli security groups can then be formed based around a job title, and those groups can then be given all the roles they need in that position.

Once configured, the administrator does not have to worry about granting access to a user to resources of many different types on many different systems. Instead, the administrator adds the user to the right Tivoli security group, and all the access rights are granted (at the next security profile distribution) through the role relationships. The time savings for

© Copyright IBM Corp. 1999

administration can be huge once TSM is in place. The problem remains that getting TSM configured and implemented is still complex.

In a small environment, the Tivoli GUI or a few Tivoli w commands are adequate for the creation of TSM security records. But, when the implementation becomes much larger, or the applications we need to secure are more complex, not only is a great deal more planning and research required, but the use of the GUI and even the command line can become repetitive and time-consuming.

In this redbook, we introduce work being conducted within the TSM product division to produce so-called *Lockdown Modules*. A lockdown module is a way of defining a security profile that can be easily altered, expanded, and, most significantly, reused in another location. With this book, we present some examples of how lockdown modules can be *ready-made* for products such as Windows NT, the Oracle RDBMS, Lotus Domino and SAP/R3. A lockdown module will include the definition of roles that provide the required level of access to perform different tasks with the subject application or operating system. Some of ours also include groups as examples of usage, but the implementation of a predefined module will involve the placement of the roles into your own group structure. The ideal TSM implementation places users in as few groups as possible, maybe even just one or two.

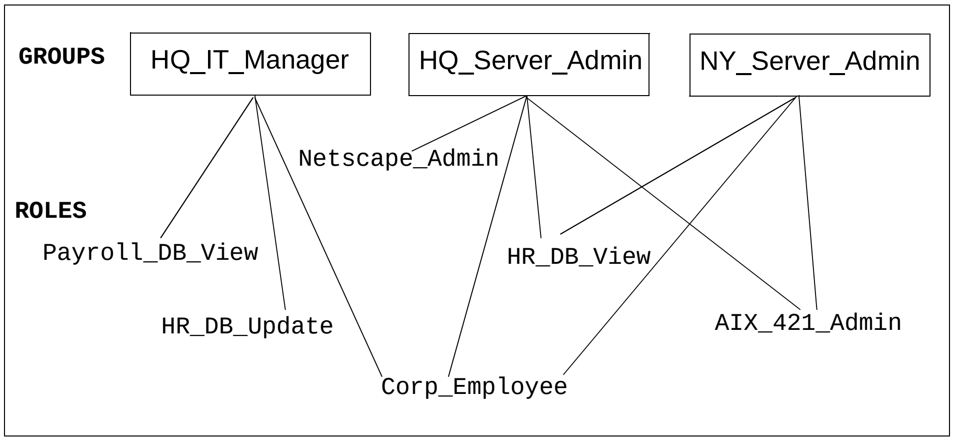


*Figure 1. Security Groups and Roles Example*

The example in Figure 1 shows how groups can represent job titles in an organization, and the tasks people perform define access rights through roles. The headquarters server administrator has access to resources similar to the New York server administrator with the addition of managing the Netscape server. The IT manager has roles that provide access to additional information (such as the payroll database) as well as greater-than typical access for the human resources database.

Besides the implementation of ready-made lockdown modules made available from Tivoli, the most important activity will be the use of the module creation process to simplify the deployment of TSM in areas specific to your environment. We present a number of examples of lockdown modules as well as details on how those modules were designed. Our discussion extends to the tools we used to review how an application worked to ensure we did not prevent them from operating once they were protected.

**Note**

Locking down applications with Tivoli Security Management should be one part of a coordinated Security Policy. While TSM greatly enhances access control management and consistency, it is of little use if other aspects such as physical access to servers or password controls are not considered.

In this book, we concentrate on the management of access control and the use of lockdown modules. You should consult the product manuals and other references regarding TSM and role-based security, building a comprehensive security policy, and understanding good TSM implementation design. This topic is addressed further in the redbook *Tivoli Security Management Design Guide*, SG24-5101.

In any security project, some balance has to be obtained between security implementation and system productivty. There is little point in spending many hours and a great deal of money to protect a resource that would only result in minor inconvenience if compromised. Figure 2 illustrates this trade-off between the decreasing risk associated with higher levels of security versus the costs involved in achieving that level of security.

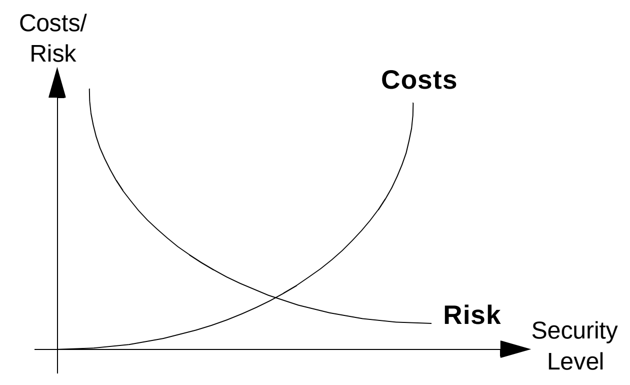


*Figure 2. Relationship Between Security Level and Costs*

As the base of lockdown modules grows, implementing extra protection for common applications will become much simpler and, therefore, cheaper to perform. Different organizations will have different views on where the optimum point on the security versus costs graph will be. Your own lockdown module design process should take this cost factor into account and aim to strike a balance in keeping with your security policy.

Part of the difficulty in defining TSM security profiles (and this is the same for any security product) is working out what resources the application uses, how it uses them, and how best to protect them. When we discuss the lockdown module examples we have tried in Chapter 4, “Lockdown Descriptions” on page 59, we try to describe how we decided on the resources to protect. In Chapter 3, “Identifying Access Requirements” on page 31, we talk about the tools we used and discuss considerations for working with each resource type such as Windows NT directories and UNIX TCP services.

The utilities provided to build lockdown modules are interesting and useful in their own right. In Chapter 2, “Introducing the Lockdown Module” on page 7, we talk about the life cycle of the security data and use the picture shown in Figure 3 on page 5.

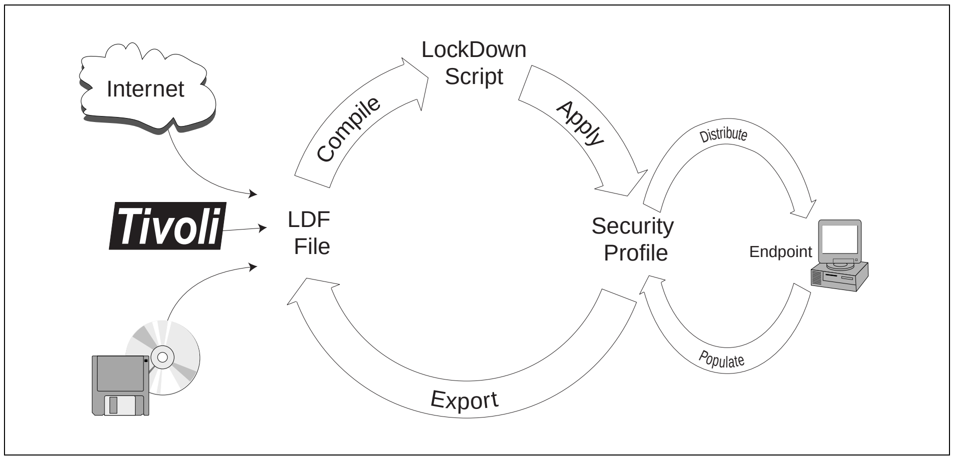


*Figure 3. Lockdown Module Life Cycle*

The security data for a lockdown module can be defined in a flat-text file called a Lockdown Definition File (LDF). We provide an LDF Import Utility (`wldfimp`) which takes that file and converts it into a script that, when run, places the security data into a profile.

We have a second tool, `wldfexp`, that has the opposite effect. It takes a security profile and exports the data into our LDF format. This can make security data much easier to read and manipulate and can provide other benefits beyond the sharing of the data such as a backup and restore path, a migration option between different environments, and so on.

## 1.1 Obtaining Lockdown Updates

The concept of Lockdown modules is very new. The intention, at the time of writing, is that there will be more modules made available as well as updates to the ones provided with this book.

Since this is such a new area, the location for updates may change, but you should start by checking the following web sites:

IBM internal network:

`http://w3.itso.ibm.com`

Follow the Additional Materials link into the ITSO Materials Repository, then select **Redpiece** and **Tivoli Security Lockdown Modules**.

Internet:

`http://www.redbooks.ibm.com`

Follow the Additional Materials link and find the SG245140 directory.

These locations are likely to change. Details will be posted in the above locations if this is the case.

# Chapter 2. Introducing the Lockdown Module

One of the most important aspects of system security is the control of who has access to what. Most operating systems have some form of access control mechanism. The capability of these mechanisms varies from one system to another. Applications often add their own access mechanisms as well as using those of the underlying operating system. For example, a database application may manage its own user accounts, but the access to files such as audit logs may be left to the access control facilities of the operating system.

Even when good access control mechanisms exist, the default configuration of a system may not be secure enough for certain environments. Different access restrictions may be needed in some cases or the default access restrictions may be too weak (allowing administrators to have unrestricted access to all resources).

Another possible concern is that operating systems and applications tend to have all-powerful user accounts that are used for executing things on behalf of the system. Users may be forced to use these accounts for maintenance purposes, and this can lead to accidental as well as deliberate security violations.

Tivoli Security Management (TSM) addresses these issues. It allows the consistent control and implementation of user access mechanisms and the deployment of system-wide security policies. Since it is such a sophisticated product, deploying TSM effectively can be tricky, and it requires a good knowledge of security, the operating system(s) and specific applications to be protected.

Lockdown Modules are one way of making a TSM deployment faster and easier. A Lockdown Module is a set of security information used to create a profile for securing a certain operating system or application. We explore the concept of the Lockdown Modules in this chapter, look at how to build your own in Chapter 3, “Identifying Access Requirements” on page 31, and describe some we generated in Chapter 4, “Lockdown Descriptions” on page 59.

## 2.1 What a Lockdown Module Does

The Lockdown Module goal is to make TSM deployment simpler and allow the exchange of knowledge and experience. A module itself consists of a number of parts. Each part represents security data in a different way.

© Copyright IBM Corp. 1999

Understanding these parts and the mechanism that converts one to another should help you to understand the concept itself.

As in many aspects of security, the lockdown modules will be improved over time. The different forms used in a module make improving and modifying them easier.

The remainder of section 2.1 explains what we mean by *locking down*, what a Lockdown Module is, why you might want to develop and design modules, and how they come to be.

**Note**

The lockdown modules provided as examples with this publication are intended to be adapted to suit your own environment. Considerable testing should be employed to ensure they operate as expected. All these examples consider the role of the UNIX root user or the NT Administrator. One aim of implementing TSM might be to restrict the capabilities of the root user. Where this hasn’t already been done in a module, great care should be taken in testing modifications to root access.

### 2.1.1 Locking Down Applications and Operating Systems

Locking down an application or operating system consists of restricting the access permissions to resources so that the system is still usable but is less susceptible to damage or unwanted access (be it accidental or deliberate).

The first goal is to avoid unwanted access to certain resources. Only people with the correct authorization should be able to access a certain resource, and the extent of this access must be controlled. A well implemented access control can avoid unwanted access or damage to information and resources.

In many environments, there are several users accessing resources in a specific system. Many of these users have to use user accounts that can potentially disrupt system operation. Also, the accounts are frequently shared by people that perform different activities in the system, some of whom may not understand all the aspects involved in managing the system. In these conditions, accidents can happen and easily cause unacceptable down times.

When locking down a system or application (or a subsystem), these goals should be considered:

- Ensure that users and the programs can operate properly and perform their activities.

- Ensure that the activities of a user do not affect other users (unless the required effect of the activity does so).
- Isolate subsystems so that administrators from one subsystem can perform their activities but not access other subsystems.
- Protect the system or subsystem from malicious external threats.

Attaining all these goals takes careful planning and time and effort. How much to restrict a system versus the cost of implementing the restriction is a business decision (see Figure 2 on page 4), but the effort can be rewarded with benefits for the whole environment.

### 2.1.2 What a Lockdown Module Is

A Lockdown Module is a system-specific or application-specific Tivoli Security Profile that can be modified to match your environment, applied on an endpoint and then tested to lockdown the specific system or application.

Building Lockdown Modules breaks down the security management task into each application or operating system that requires protection. Each module defines Tivoli security roles that can be given to groups to enable them to perform tasks that would otherwise be restricted. Tivoli security groups and roles are one area of the module that is likely to require some customization to fit into your security model.

### 2.1.3 Why Use a Lockdown Module?

The chief advantage of Lockdown Modules is that they are easy to use. Being modular makes them easy to distribute, edit and manipulate. Well designed modules can be applied together so that different aspects of the system are addressed by each module, and the overall result is a fully-locked target.

Since modules are designed for specific components, they can easily be handled independently. So, a module can be designed, tested, and altered on its own, and then applied to its final environment.

Another interesting aspect of Lockdown Modules is that they can be represented in a number of different formats. By exporting an existing security profile into a Lockdown Definition File (LDF), it is possible to study and modify a module isolated from the TMR environment. This also allows the modules to be sent to other people and organizations.

Exchanging modules is a good way of checking and validating your profiles. As in many other areas, knowledge of the application or system is the key for effective use of the security tools. As Lockdown Modules are tested, it is recommended that updates and improvements be shared since this results in better overall modules. See 1.1, “Obtaining Lockdown Updates” on page 5 for information about updates to those modules already provided as well as any new ones that may become available.

### 2.1.4 Life Cycle of a Lockdown Module

Much like applications, Lockdown Modules have a life cycle of their own. Figure 4 shows this cycle.

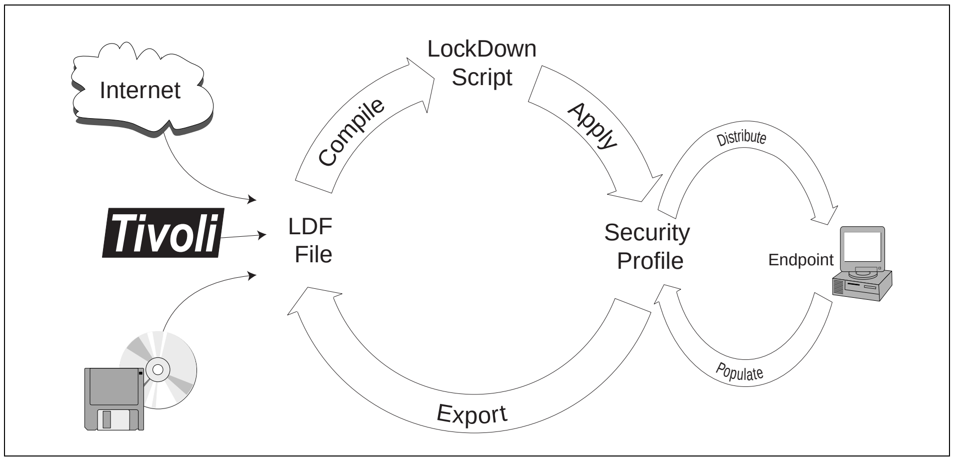


*Figure 4. Lockdown Module Life Cycle*

Lockdown Module security data can exist in three forms:

- LDF File
- Lockdown Script
- Security Profile

There are ways of transforming the Lockdown from one format to another. The Security Profile is the final form. LDF files and Lockdown Scripts are the means to get the profile created. LDF files can be developed by hand or exported from an existing Tivoli security profile. (See 2.2.6, “Security Profile Export Tool” on page 26.)

Note that, at different stages in the life cycle, we can modify the security data in different ways. We can use the LDF file format as an easy way to add large numbers of resources or role definitions, and use the security profile in TSM for modifying group memberships and making more incremental changes. There may even be occasions when it is useful to manipulate the scripts that will generate security profiles so that we can perform more sophisticated functions.

Lockdown modules can come from various sources. They will mostly be created out of necessity within your own environment. Lockdown modules provide a mechanism of exchange between departments or between development, test and production environments. With time, it will be possible to obtain them elsewhere. See 1.1, “Obtaining Lockdown Updates” on page 5 for information about updates to those modules already provided as well as any new ones that may become available. Lockdown modules provide a method for backing up and restoring security definitions. This method may be preferable to using other options such as the Tivoli Migration tools since lockdown modules are specific to security data.

**Important**

Lockdown Modules need thorough testing and validation. As good as any module may be, it will almost certainly need modification to ensure it is adequate and safe in your environment. Always experiment with the modules in a stand-alone test environment. For UNIX targets, we recommend the use of TACF Warning mode during testing.

#### 2.1.4.1 Lockdown Definition File

The Lockdown Definition File (LDF) is in human-readable and editable file format. It can be used to exchange Lockdown Modules and edit them. It can be edited from scratch or obtained by exporting an existing security profile using Tivoli migration tools or the export utility we provide (see 2.2.6, “Security Profile Export Tool” on page 26). The LDF has a simple format and is easy to build. Any editor that saves the file as plain text can be used to edit the LDF file.

Another way of editing a Lockdown module is by changing the security profile. This can be done either through the Desktop or by using the CLI commands. The profile can then be exported to LDF format if required. Updates are very likely to happen during the life of a Lockdown Module.

#### 2.1.4.2 LDF Import Utility and Lockdown Script

The LDF import utility we provide (see 2.2.5, “LDF Import Utility” on page 23) is used to transform an LDF into a shell script that can be run on UNIX or in the Tivoli Windows NT bash environment to build a security profile. It does a series of syntax checks and generates the script that will create the Lockdown Module’s security profile. The script must be executed in the TMR

Server or in a managed node that contains the TSM commands. Running the script itself will not affect the security of your systems since it only creates the profile. However, the profile should be thoroughly checked before distributing it. As we have stated before, on UNIX (with TACF) we recommend the use of Warning mode before fully implementing a profile.

**Note**

The script that is built from the LDF will delete the named profile if it already exists in the named profile manager. Always check to ensure that the names provided in the LDF and subsequent script will not result in the deletion of a required profile. This behavior is determined by the ldfc.pre file (also known as the preface).

As mentioned previously, there is an export tool supplied with the import utility. It converts an existing security profile into an LDF file. This file can then be edited and reapplied, or, it can be exchanged with other people. It is hoped that, through constant exchange and updates, Lockdown Modules can be improved and new ones introduced.

## 2.2 Building an LDF for an Application

A Lockdown module is essentially the security profile. Thus, it is a set of security records that deal with Tivoli security *Resources*, *Roles* and *Groups*. We assume, in this publication, that the reader is familiar with TSM Security Profiles.

In this section, we describe in more detail the Lockdown Definition File (LDF). The resulting security profile itself is no different than the normal security profiles. The LDF, however, is a little different from the normal formats (available through the Desktop or the `wlssec` command).

The LDF format will be described and demonstrated using a hypothetical application. The Lockdown Module concept will make more sense when shown in a specific context. Real applications can be very complex to secure and would be too lengthy for an example such as this. Once you are familiar with the LDF concepts, you can review the samples provided with this publication (see Chapter 4, “Lockdown Descriptions” on page 59). For a formal definition of the file format, see Appendix B, “Lockdown Definition File Format” on page 105.

### 2.2.1 A Hypothetical Application

This section will describe how you decide to secure an application by using a hypothetical example.

HYPAPP is a UNIX application that works like many Client/Server applications. Clients connect to the server and make requests of it to perform an activity. There is an application-specific user authentication process, but since we do not manage such authentication through TSM, this is not relevant to the LDF. Here is a list of the main characteristics of our HYPAPP application:

- It can be installed in any directory or mount point. This initial directory becomes the application’s home directory. Inside the home directory (<home dir>) it creates these directories:

  | | |
  |---|---|
  | <home dir>/bin | Contains all the binaries for the application. |
  | <home dir>/bin/cfg | Contains a series of configuration files used by the client programs. |
  | <home dir>/data | Contains the data files used by the server, the contents of this directory should only be updated by the server process. |
  | <home dir>/dump | Contains the dump of the data files. These are used for backup and restore operations. |

- All activities in the database directory and the dump directory are carried out by a daemon named hypappd. This process also handles user requests. If it is killed with transactions open, the consistency of the database may be compromised. To start and stop the daemon, there is a special program call `hypctrl`. This program has internal authentication and uses a service TCP port to communicate with the daemon. The daemon runs under a specific account using the user name hypapp.
- Several maintenance and operational activities are required for the application. These activities can be identified as the following roles:

  | | |
  |---|---|
  | Operation | This involves starting and stopping the server. Other activities are conducted internally through the application’s binaries. |
  | Administration | Besides regular operator tasks, there are some configuration and maintenance tasks that may be performed. These include maintaining dump files, configuring the server, and updating binaries. |
  | Backup | A snapshot of transactions and the database are sent to the dump directory and, from there, must be transferred to another device (normally tape). |

| | |
|---|---|
| Normal Use | Most activity is conducted by a user that accesses the client part of the application and makes requests of the server for specific activities. |

- Since there is an internal authentication built-in to the server operation, any user can access the client binaries and attempt to connect to the server.

The knowledge of how an application works is crucial for the creation of any Lockdown Module. In the examples that follow, this sample application will be used to provide matter and context to be added to the Lockdown. It will also be used in the LDF examples.

### 2.2.2 Determining the Scope of the Lockdown Module

In order to design a Lockdown Module, it is necessary to set the goals for it. Different environments will demand very different goals. In this example, the Lockdown Module was designed to do the following:

- The UNIX root user should not be able to interfere in the operation of the applications.
- Accessing the daemon process and starting it must be restricted.
- Remote shutdown of the application through the service port must be disabled.
- Backup operation must occur without problem, but only authorized operators can be allowed to read from the <home dir>/dump directory.
- The <home dir>/data directory must be protected from all users; only the application’s daemon should access it. This includes preventing the operators and administrators from writing to the files.

### 2.2.3 Hypothetical Application Lockdown Definition File

This section contains the LDF used to generate the Lockdown Module for out hypothetical application, HYPAPP. The whole file is shown split into fragments that are explained individually. Comments on what is contained in each fragment is provided after each one.

This first fragment contains the Lockdown name and variable declarations (see Figure 5).

```
# This is a LockDown Module for the HYPAPP application
#
# This section informs the import utility of the LockDown Name
LOCKDOWN HypApp_LockDown

#
# Variable declarations
#
VAR
  HYPAPPHOME=/usr/local/hypapp;
  HYPAPPBIN=${HYPAPPHOME}/bin;
  HYPAPPDB=${HYPAPPHOME}/data;
  HYPAPPDUMP=${HYPAPPHOME}/dump;
  HYPAPPCONFIG=${HYPAPPBIN}/cfg;
  HYPAPPD=${HYPAPPBIN}/hypappd;
```

*Figure 5. HYPAPP LDF - Name and Variable Section*

The name is the word after the keyword `LOCKDOWN` (separated by a space). It is used as the name for the security profile, profile manager, and the output script. Tivoli recommends a naming convention for Tivoli objects that helps identify the object type and function. For example, a security profile will have a name that describes what it secures and may have a suffix of _SP to identify it as a security profile. The use of the `LOCKDOWN` variable in names of security profiles and profile managers is determined by the Import Utility preface file (ldfc.pre) and this may need some modification to build scripts that are consistent with your naming convention.

**Note**

When the resulting script is run, it will attempt to delete a profile manager and profile of the name specified by `LOCKDOWN` if they already exist. If the same name is used in a profile of a different name, it will not be affected.

The other element present in Figure 5 is the variable declaration block, identified by the `VAR` heading. Everything from this heading to the next reserved heading name will be interpreted as a variable declaration. In this case, it is being used to declare the directory names to be used in a later section of the LDF (the resource section). The main directory is HYPAPPHOME representing the <home dir> mentioned previously. There are several other directories that are subdirectories of this one. This is reflected by the `${HYPAPPHOME}` in the value of the variable. This indicates that the import utility must expand the HYPAPPHOME variable in these names. This happens upon declaration of a variable or resource.

Using variables eases maintenance and allows more flexible lockdown modules. Values for variables can be set when the import utility is invoked.

Figure 6 shows how Tivoli security groups are declared in the LDF.

```
#
# Group Declaration section
#
GROUPS
  HAG_Admin "HYPAPP Administators" = (haadmin) audit(LA,RF);
  HAG_Backup "HYPAPP Backup Operators" = (haoper) audit(LA,RA);
  HAG_Operator "HYPAPP Server Operators" = (haoper) audit(LA,RA);
  HAG_Deamon "Deamon Group" = (hypapp) audit(LA,RN);
```

*Figure 6. HYPAPP LDF - Group Section*

Groups declared here can be referred to in the LDF roles section. The group names are followed by a quoted string. This string is an optional description or comment for that group. This kind of construction is also used in other declarations.

The name or comma-separated names optionally placed in the parentheses immediately following the equal sign are names of the users that belong to the group. After the names of group members, we can specify TSM properties. In this example, they are setting the level of auditing that is needed for these groups. For the Backup and Server Operator groups, all access and logins are logged. For the daemon, all logins are logged but not resource access. And, for the administrators group, all logins and resource denials are logged. See B.2.4, “LDF GROUPS Section” on page 114 for more information.

**Note**

The groups we define in the sample LDFs in this publication are to provide examples of suggested use of roles. In your environment you will probably not use these groups. Instead, you will add the roles that are created to your own security group structure.

Figure 7 shows the roles defined in the HYPAPP LDF.

```
#
# Role Declaration Section
#
ROLES
  HAR_Backup "Backup Role" = (HAG_Backup);
  HAR_Admin "Adminatrator Role" = (HAG_Admin) parent(HAR_Operator);
  HAR_Operator "Server Operator Role" = (HAG_Operator) ;
  HAR_Deamon "Role of Deamon, for the deamon only" = (HAG_Deamon);
```

*Figure 7. HYPAPP LDF - Role Section*

These roles will later be used in the LDF access declaration section. The parentheses after the equal sign can optionally contain the names of the groups that have this role (they must have previously been declared in the group section). To work with groups already existing in other profiles, we must use other means such as the pass attribute or the PROLOG and EPILOG sections - See Appendix B, “Lockdown Definition File Format” on page 105 for more information). Following the group associations, we can specify other attributes. One of the roles has a parent role. This is indicated by the parameter parent and the name in parentheses is that of the parent role. See B.2.5, “LDF ROLES Section” on page 114 for more information.

After having declared roles and groups, the LDF contains the resource declaration (shown in Figure 8).

```
#
# Resource Declaration Section
#

RESOURCES
  ${HYPAPPHOME}/* "HYPAPP base dir" = [R];
  ${HYPAPPBIN}/* "HYPAPP Binaries" = [RX];
  ${HYPAPPDB}/* "HYPAPP Data Files" = [N] audit(RF);
  ${HYPAPPDUMP}/* "HYPAPP Backup" = [N];
  ${HYPAPPCONFIG}/* "HYPAPP Config Files" = [R];
  FILE:${HYPAPPBIN}/hypctrl "HYPAPP Server Control Program" = [N];
  FILE:${HYPAPPD} "HYPAPP Deamon" = [N];
  PROGRAM:${HYPAPPD} "HYPAPP Deamon" = [N];
  PROCESS:${HYPAPPD} "HYPAPP Deamon Process" = [N];
  TCP:hpctrlport "HYPAPP Control TCP port" = [N] tcpaccess(+localhost);
```

*Figure 8. HYPAPP LDF - Resource Declaration Section*

This declaration sets the default access to the resources (later modified for roles in the ACCESS section). The access is defined by the standard TSM letters enclosed in square brackets ([]).

The letters TSM uses to denote access can be found in the product documentation in the *TME 10 Security Management User's Guide* appendix on Resource Types.

For this specific case, this declaration defines:

- Read access to the home directory of the application.
- Read and execute access to the binaries.
- No access to the data files, and there is also auditing on resource access failure for these files.
- No access to the backup files.
- Read access to configuration files.
- The files `hypappd` and `hypctrl` (daemon program and server control program respectively), are not accessible.
- The daemon process is also protected (the `PROCESS` line).
- The control port used to communicate to the daemon is also protected, no access is granted except from the localhost (indicated by `tcpaccess(+localhost)`).

All these declarations set the basic permission restrictions that are the goal of the lockdown module. All that needs to be done now is to grant specific access to resources for each role (see Figure 9).

```
#
# Access Declaration Section
#

ACCESS

ROLE HAR_Backup =
  ${HYPAPPDUMP}/* [R];

ROLE HAR_Operator =
  ${HYPAPPBIN}/hypctrl [RX],
  FILE:${HYPAPPD} ${HYPAPPBIN}/hypctrl[X];

ROLE HAR_Deamon =
  PROGRAM:${HYPAPPD} [X],
  ${HYPAPPDB}/* ${HYPAPPD}[F],
  ${HYPAPPDUMP}/* ${HYPAPPD}[F],
  ${HYPAPPHOME}/* ${HYPAPPD}[F];

ROLE HAR_Admin =
  ${HYPAPPDB}/* [RX],
  ${HYPAPPBIN}/* [F],
  ${HYPAPPHOME}/* [F],
  ${HYPAPPCONFIG}/* [F];
```

*Figure 9. HYPAPP LDF - Access Declaration Section*

Each little block starting with `ROLE` describes the specific access a certain role has to resources. The resources and roles were all declared previously. Again, the access permission is indicated by the letters enclosed by square brackets, however, there is a special construction shown in this example:

```
${HYPAPPDB}/* ${HYPAPPD}[F]
```

This statement indicates that the permission is a conditional access. In the example above, it means that full control `[F]` to the database directory (`${HYPAPPDB}/*`) is granted for the role using the daemon program (`${HYPAPPD}`). This type of access is extremely useful when locking down UNIX systems and applications.

The grants shown in this segment will result in the following:

- HAR_Backup can read from the dump directory (the default is no access).

- HAR_Operator can start and stop the server. This is done using the `hypctrl` program and the `hypappd` program, both otherwise not accessible.
- HAR_Deamon is used to grant conditional access to the program and, while using it, have full control over all the other directories. This role is only used to enable the daemon to do its job, it should not be granted to real users.
- HAR_Admin can read and execute in the database directory and has full control over binaries, configuration files, and the home directory.

This completes the LDF file. The resulting Security Profile implements the goals desired for this Lockdown Module.

### 2.2.4 Complete Lockdown Definition File Structure

The example in the previous section shows how to conceive and implement a Lockdown module. The notation showed examples of how to specify TSM entities in an LDF. The example did include almost all the possible elements of an LDF. A typical file will contain all the sections used in the HYPAPP LDF example.

The LDF import utility is composed of seven sections:

1. Variable Declaration (VAR)
2. Prolog (PROLOG)
3. Group Declaration (GROUPS)
4. Role Declaration (ROLES)
5. Resource Declaration (RESOURCES)
6. Access Declaration (ACCESS)
7. Epilog (EPILOG)

Each of these sections is optional but the order is significant and cannot be changed. Sections 3 through 6 are used to create the records in the security profile. References to groups, roles, and resources within the LDF must be matched by a previous declaration. The use of existing TSM resources, groups, and so on is possible by adding to the import script through the use of the PROLOG and EPILOG sections or by using the pass parameter to pass switches to the script commands.

**Note**

The resulting import script will also include the ldfc.pre preface file.

#### 2.2.4.1 Variable Declaration

This section is used to declare variables that will be expanded when found in other declarations. This expansion occurs at conversion time and at the moment in which the variable is found.

Variables can be used to enable LDF files to be able to handle instance specific data such as the mount point, instance name, host names, and so on. The value can be defined in the LDF file or in the arguments that are passed to the import utility. If a variable used in a declaration is not found in the variable declaration block, or is not supplied to the import utility, the import utility will not accept the file.

It is possible to use the korn shell (ksh) variable notation for variables that should be expanded during the script’s execution, that is, after the LDF has been converted to a script and when that script is executed to build the security profile. This also allows variables from the ksh environment to be handled; the import utility does not modify them. The variables must exist or be declared in the PROLOG section. The variable values must be valid for the record type. Note that the variables are not stored in the security profile and so are not used by TSM when the profile is distributed - the actual values of the variables current at the time the import script was run will have been stored in the profile.

**Note**

We can use two types of variables in the LDF. Those of the form ${VARIABLE} must be defined in the LDF VAR section and are evaluated by the import utility as it builds the import script. Those of the form $VARIABLE are considered ksh variables by the import utility and are placed into the import script as ksh variables. When the import script is run to build the profile, the shell evaluates the variables. In this way, we can specify variables in the LDF to improve readability and make alterations easier, and we can use shell variables (for example in prolog scripts) to add functionality to the LDF based on data that should be gathered at the time the import script is run.

For more information, see B.2.2, “LDF VAR section” on page 112.

#### 2.2.4.2 Prolog

Prolog is used to include scripts that must be executed before the Security Profile is created. This may be used for maintenance or to create information that is to be used by the LDF.

The script is included in the resulting script by the import utility and should be a valid ksh script. If the LDF variables exist in the prolog script, they are expanded. The import utility does not check for syntax or semantics in the included prolog script. There is no need in the prolog script to create the actual profile that will hold the lockdown module since this is created prior to the prolog section. It is possible to modify the lockdown profile from the prolog, but this may cause problems in the execution of the rest of the script.

For more information, see Appendix B.2.3, “LDF PROLOG Section” on page 113.

#### 2.2.4.3 Groups Declaration

The Groups Declaration section specifies which groups will be created in the Security Profile and, optionally, membership and other parameters for the group. The groups are usually assigned to roles. There is no checking for the existence of users named in the groups.

It is possible to use references to users stored in Tivoli user profiles from Tivoli User Administration; again, there is no checking on the correctness of this user data. If the user does not exist, the script will fail. User profile references must always be quoted.

See B.2.4, “LDF GROUPS Section” on page 114.

#### 2.2.4.4 Roles Declaration

TSM role records are created based on the declarations contained in this section. It is also possible to assign groups to the roles using this section. The specific access rights for a role are declared in the Access declaration section. See 2.2.4.6, “Access Declaration” on page 23.

The declarations in this section are used only to define the role and role properties, not the access permissions. Parent roles can be declared, but, as in other sections, all parent roles and groups must be declared in the same LDF file.

For more information, see B.2.5, “LDF ROLES Section” on page 114.

#### 2.2.4.5 Resource Declaration

The TSM resource records are created based on the information declared in this section. This section also declares the resources that can be placed in role-specific access declarations.

Default access is specified in this section. Other TSM resource properties can also be set in this section.

For more information, see B.2.6, “LDF RESOURCES Section” on page 115.

#### 2.2.4.6 Access Declaration

In the previous sections, roles, groups, and resources have been declared. The last element that needs to be dealt with is the access that the roles will have to resources. This is the purpose of this section. We separated this from the declaration of the roles themselves to make the LDF file more readable and more simply constructed.

Each statement can refer to a resource or a role. When it refers to a resource, it is followed by a list of all roles and the access those roles have to the resource. The other form refers to a role and then lists the resources that the named role has access to. Both forms can be mixed and are equivalent. Conditional access can also be granted.

All elements being referred to in this section must have been previously declared, otherwise, the import utility will not accept the LDF file.

For more information, see Appendix B.2.7, “LDF ACCESS Section” on page 115.

#### 2.2.4.7 Epilog

There are some things you may wish to do that are not part of the current LDF import utility such as things relating to the fact that it requires every resource, role and group to be declared in the LDF file. Another problem is that it does not handle Tivoli User Administration (TUA) records. So, it may be required to do some changes to the profile after it has been created.

The Epilog section is designed for this purpose. Its content and behavior is the same as the Prolog section, but the includes are added to the script after the Lockdown profile script has been created. This section is used for updates and changes to the profile, such as adding resources from other Security Profiles, integration with TUA, adding subscribers to the profile manager, and so on.

The same restrictions and considerations for the prolog section apply to this section. For more information, see Appendix B.2.3, “LDF PROLOG Section” on page 113.

### 2.2.5 LDF Import Utility

In order to import an LDF file into a security profile it must be converted into a script. The result of this conversion is the *Lockdown Script*. This script will create the lockdown module's security profile. Conversion is the responsibility of the LDF Import Utility and it is a simple process.

#### 2.2.5.1 LDF Import Utility Components

There are two components for the import utility. The import script converter itself, named `wldfimp` and the preface file named `ldfc.pre`. The converter is an interpreter-specific binary. At the time of this writing, versions were available for Solaris, AIX and Windows NT, with others to follow.

The preface is a ksh template used to generate the necessary environment for the execution of the script. It is included at the beginning of the script and ensures that the profile manager and security profile are validly specified in the LDF. Other preface files can be used, assuming they ensure the necessary environment exists for the remainder of the script (see "Script Preface" on page 27).

#### 2.2.5.2 Using the LDF Import Utility

The converter can be invoked in the following manner:

```
wldfimp lockdownldffile.ldf
```

The converter will read the file and check it for syntax and correct use of options. Error messages are shown in this format:

```
W(11):Attempt to redefine variable, ignoring redefinition
E(48):Endpoint NT does not support resource type PROGRAM.
E(49):Endpoint NT does not support resource type PROCESS.
```

The `E` stands for error and the `W` for warning. The number in parenthesis is the line number that is incorrect (unless the error is concerning a role parent declaration, since these are checked at the end of the file).

An LDF is designed for one type of endpoint, and attempting to build scripts for another interpreter type will cause errors. To specify which endpoint the converter should assume for the LDF file, use the -nt or -ux flag (the default is that of the interpreter it is run from).

This will convert the LDF file into a script that will import the data into a security profile. There are several options available; to see them use:

```
wldfimp -?
```

One useful option is upper case D (`-D`). It allows variables to be defined or redefined. The definition comes prior to that of the Variable (VAR) block in the LDF. If the variable is redefined, the first definition is kept, and a warning is issued.

```
wldfimp -D HYPAPPHOME=/hypapp hypapp.ldf
```

Variable expansion also happens in these definitions; so, it is possible to declare them as follows:

```
wldfimp -D HYPAPPHOME=/usr/local/hypapp -D ‘HYPADPPBIN=${HYPAPPHOME}/bin2’ hypapp.ldf
```

The single quotes (‘) are used to avoid ksh from trying to replace the variable. Both values must be declared because there is no prior declaration of `HYPAPPHOME;` so, it must also be declared.

For information on the other options, see Appendix A.2, “The LDF Import Utility” on page 103.

### 2.2.5.3 LDF Import Utility Tricks and Tips

One important detail about the converter and output script operation is that variables are handled twice. The first time is during the conversion of the LDF file. The variables in the LDF format are replaced by their value. The next one occurs when scripts replace the variables.

The first one handles variables in the format ${VARIABLE}, and the second handles those in $VARIABLE format. Thus, it is possible to handle some variables in the LDF and leave others to be handled by the script.

Variables handled by the script are treated just like normal ksh variables. They must be set either in the ksh environment or in the prolog scripts. This can be particularly useful when some of the information needed must be gathered when the script is executed - such as a list of managed nodes.

For instance, when locking down the TMR, it is possible to use variables such as $BINDIR, $DBDIR, and so on. Remember, however, that these variables are interpreted when the script is executed, and they are not stored in the profile. You cannot specify variables that are interpreted once the profile reaches a target when the profile is distributed.

One way of getting the values in the variable is by using look-aside files. The files contain text that will be turned into the values and are read by the prolog. So, it is possible to rerun the script when new values are set for these files and then distribute the new profile. Careful building of prolog and look-aside files can make the tool very adaptable and powerful.

When exporting a profile, it may be advisable to keep the replacement variables in a file. This makes it easier to reuse those variables if the profile is exported again - an exported profile will not generate any variables without them being defined in a file. The export tool we provide can replace strings with variable definitions as described in the next section.

### 2.2.6 Security Profile Export Tool

Since the Desktop will often be the starting point for the definition of Lockdown Modules, it is fundamental to have some means of exporting the existing profiles.

The `wldfexp` does exactly that. It takes an existing profile and creates the LDF that would recreate it. The resulting LDF can be edited and reconverted to produce the import script that will create the profile again.

Some editing may be necessary to make the LDF exchangeable. The export tool already handles variable replacement in much the same way the import utility does. But, a truly portable LDF will need some editing after it has been exported.

To export a profile, issue the following command:

```
wldfexp HypApp_LockDown
```

`HypApp_LockDown` is the name of the security profile, and the export tool will generate a file of this name with an `ldf` extension (in the example above `HypApp_LockDown.ldf`). There are several other options that affect the general output of the export tool.

One very important option is `-V`. This will create a variable definition and allows the export tool to replace text in the name of a resource to its matching variable. For example:

```
wldfexp -V '${HYPAPPHOME}=/usr/local/hypapp' HypApp_LockDown
```

Each time the `/usr/local/hypapp` text is found, it will be replaced by `${HYPAPPHOME}`. This allows the export to recreate the original LDF or to make the LDF easier to change. It is possible to place all the variable replacements in a file and use the `-v` option to load it. For example:

```
wldfexp -v hypapp.ldv HypApp_LockDown
```

Other very important options are the `-nt` and `-ux`. These specify what type of module is being created. The LDF import utility provided with this book has a specific version for each type of endpoint. Choose the correct one when exporting a profile.

For more details on the option for the export tool see Appendix A.1, “The LDF Export Tool” on page 101.

## 2.2.7 Lockdown Script

The result of the import utility conversion of an LDF is a Lockdown script. This script is used to generate the security profile for the Lockdown Module. A lockdown script is composed of four parts:

1. Preface
2. Prolog
3. Lockdown Definition
4. Epilog

### 2.2.7.1 Script Preface

The preface is a standard profile that is used to create the necessary infrastructure for the creation of the Lockdown Module on the TMR. There is a default preface script, but this can be changed.

After its execution the following conditions must be met:

- The profile manager for the security profile must exist. It's name is given by the $PRFMGR variable.
- The security profile must exist. The name is given by the content of the $PROFILE variable.
- The security profile must be empty.

So, if you wish to change the preface, you must ensure that these conditions are met. It is also recommended that the security profile reside in its own profile manager since some security records must be unique within the profile manager. Variable references in the LDF format (`${VARNAME}`) are expanded in the preface file prior to its inclusion in the Lockdown script.

### 2.2.7.2 Script Prolog

The prolog comes after the preface, hence, when the security profile already exists but is empty. It can be used to set variables that must be used in subsequent parts, or to ensure other preconditions to the rest of the script.

**Note**

The prolog should not change the $PROFILE variable or populate the profile since this can make the remainder of the script invalid.

A prolog is optional in the LDF script file, and this section only exists if there is a prolog section in the LDF. Subsequent changes to the prolog files are not reflected in the Lockdown script since the script includes the whole prolog file. Variables in the LDF format are also expanded in the prolog files. The import utility does not check the includes for syntax.

#### 2.2.7.3 Script Lockdown Definition

In the Lockdown definition part of the Lockdown script, the security profile records are generated. The import utility tries to analyze and validate all the input for this part so that it works properly, however, errors may occur. Some options in the LDF allow the writer of the LDF to pass arguments that may be incorrect.

This part creates the groups first, then the resources, and finally the roles. Failures can propagate from one part to the other, that is, a role may fail to be created because a resource referred to by it does not exist. When this occurs, it may be necessary to analyze the script and verify the command that failed. If you do not receive errors, there is normally no reason for changing or studying this part.

#### 2.2.7.4 Script Epilog

The epilog part is very similar to the prolog part, but it is executed after the profile records have been generated. It can be very useful for modifying the profile in other ways to handle options that are not dealt with by the LDF import utility.

Unlike the prolog, anything can be done in the epilog: changes to the profile and to variables, changes to the profile manager, and so on. Variables in LDF format are converted to their contents in the security profile and all variables declared in the preface and prolog are available.

The main purpose of the epilog is to allow a lockdown to perform activities that the import utility does not handle. The epilog is optional.

## 2.3 Additional Uses for Lockdown Modules

There are other ways in which lockdown modules can be used to build an effective TSM implementation. Examples include the subdivision of root capabilities in UNIX. Typically, we would build up the necessary data in a security profile and use `wldfexp` to export that to an LDF file. In LDF format, it can be modified and extended, and prolog and epilog scripts can be added.

Tivoli provides a number of tools related to backup, such as the `wbkupdb` command and equivalent GUI operation and utilities such as those provided in the migration toolkit. However, there may be other times when you need a solution specific to TSM. These utilities allow you to create archives of security configuration data. In a very secure environment, it may be very important to know what the security configuration was at any particular moment in time. If the platform security model does not keep track of this type of information, you can save it from profiles after each set of changes. Obviously, there can be differences between what is in a Tivoli security profile and what is implemented on a system, but this may be an option for maintaining such records as part of a larger auditing plan.

# Chapter 3. Identifying Access Requirements

In this chapter, we describe the process required to identify what application resources we can protect through a lockdown module.

When we look at what to protect, we need to understand not only what files and other resources we are dealing with, but who we are to protect them from. For example, it would be good practice to prevent the superuser (such as a root ID on UNIX or Administrator in Windows NT) from modifying log files, especially log files that contain auditing information.

With the Tivoli Access Control Facility (TACF) on UNIX, we can treat the root user in the same way as any other and thus secure the system or an application against accidental or deliberate damage. For Windows NT, we can define the access control on a file such that the Administrator cannot access it; but, remember that the Administrator ID can always grant itself the access again. In Windows NT, the aim should be to maintain the minimum number of Administrators and to have other operator functions managed through NT user rights and other access controls.

## 3.1 UNIX and Windows NT Security

There are major differences in the security management capability we have between the UNIX platforms and Windows NT. Due to the fact that different versions of UNIX implement different security models, Tivoli uses the Tivoli Access Control Facility (TACF) to provide a consistent security interface for all supported UNIX platforms. TACF maintains its own security database to check access permissions and allows us to treat all users, including root, in exactly the same way. TACF implements TSM's role-based security model through Tivoli security groups and roles and provides a consistent access control list (ACL) capability. TACF also enables us to protect more resource types than is typically possible in a UNIX security model and more than we can protect under Windows NT security including TCP service access, for example.

When securing resources in UNIX, it is important to remember that, even if TACF grants the requested access, the request still has to pass through the regular UNIX security check before completing. If, for example, native UNIX permissions do not allow access to a file, and the TACF daemon (seosd) does allow Read, TACF passes the request, but UNIX returns access denied. TACF is always checked first, then the request is passed on to UNIX.

The TSM implementation for managing Windows NT does not use a second layer like this. Instead, since Windows NT has a security model that maps well to TSM's role-based security model and already uses an ACL architecture, TSM merely applies accesses and roles to its resources using the native Windows NT security API.

This improves productivity and consistency for administrators in a number of ways. All platform types (including those that can be added to the base product such as OS/390 and OS/400) can be managed using the same model and using the same graphical and command line interfaces. The ability to define security relationships between people and resources and apply them across many different systems can save plenty of time over native methods such as applying security properties and permissions on each file and directory. Natively, this would involve right clicking on each file or directory, bringing up its properties, bringing up the security permissions, changing permissions, and applying them for each system, even if the same resource was being protected on many different systems. TSM automates the process by applying the permissions by roles and groups in one profile distribution.

Note that in TSM versions up to and including 3.6, there is an exception in handling Windows NT files beneath a directory. In 3.6, we do not support applying a directory's permissions to the files within it, unless those files are individually defined in TSM resource records with a default permission (which is not recommended). From 3.6.1 onward, not only can we define access rights to all existing files in a directory with wildcards, but we can also apply directory permissions to all subordinate directories.

## 3.2 Tools to Identify What to Protect

Finding out how an application is operating can be a complex task. This section presents some suggestions for making that task a little easier.

### 3.2.1 Watching Windows NT Applications

Setting up resources and their access properties can be time consuming and difficult if having to use trial and error. Either access to a resource is set to be too restrictive, and productivity declines, or it is set too weak, and security can be compromised. In order to help determine which resources we needed to add in our designs and the accesses that were minimally required, several tools were used. TSM's TACF has a trace facility (see 3.2.2, "Watching UNIX with the TACF Trace" on page 37) which causes the TACF daemon (seosd) to output messages that report its operations and actions to a trace file or a console. TACF also has the ability to run in a "Warning Mode" to show access rule decisions without having to actually apply them. Unfortunately, no such tool exists within Windows NT; so, we looked for third party tools that were available on the Internet. The following tools are described here:

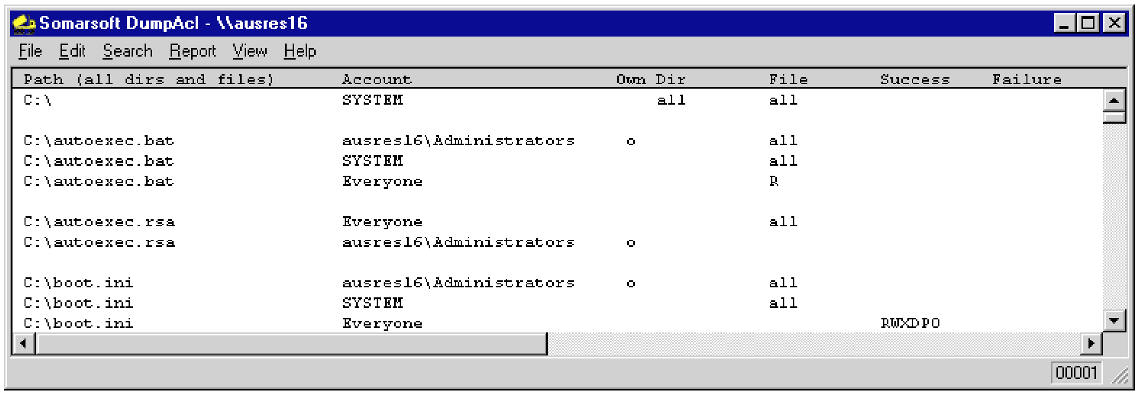
| | |
|---|---|
| **Dumpacl** | Windows NT ACL information tool |
| **Filemon** | Track file system activity |
| **HandleEx** | Determine activities of processes |

**Note**

Before using these tools, be sure to check the licensing agreements. Most shareware and freeware products require registration when used for commercial purposes.

### 3.2.1.1 Dumpacl

This utility from `http://www.somarsoft.com` can be used if you need a thorough examination of Windows NT resources; we encourage you to use this or a similar tool to help understand the security configuration of resources. This utility can scan entire drives for file, directory permissions, registry, printers and share information. Dumpacl also dumps user, group, and replication information.


Somarsoft DumpAcl - \\ausres16
File Edit Search Report View Help
00001

```
Path (all dirs and files)      Account                    Own Dir     File        Success     Failure
C:\                            SYSTEM                         all     all

C:\autoexec.bat                ausres16\Administrators     o          all
C:\autoexec.bat                SYSTEM                                 all
C:\autoexec.bat                Everyone                               R

C:\autoexec.rsa                Everyone                               all
C:\autoexec.rsa                ausres16\Administrators     o

C:\boot.ini                    ausres16\Administrators     o          all
C:\boot.ini                    SYSTEM                                 all
C:\boot.ini                    Everyone                                           RWXDPO
```

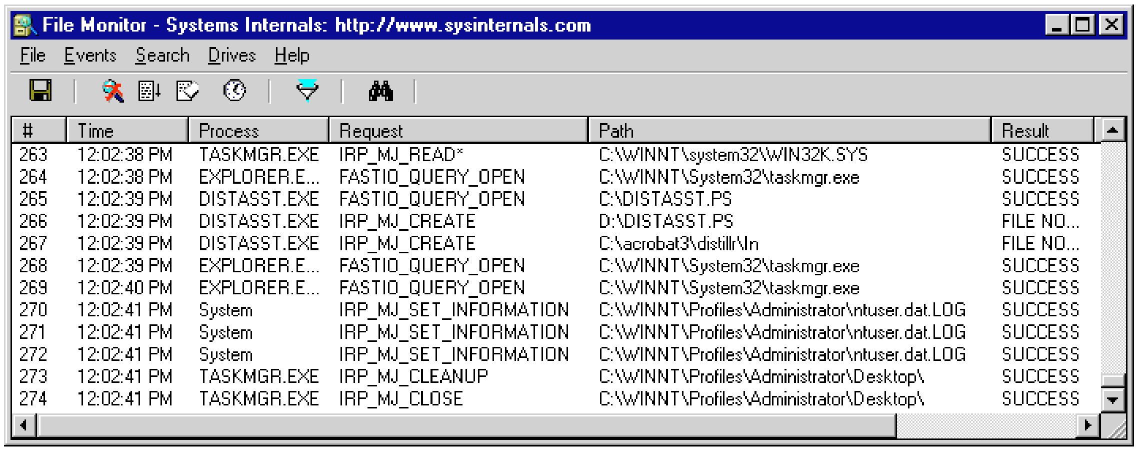
*Figure 10. Dumpacl - File and Directory Sample Output*

As you can see from Figure 10, the output is in an easy-to-read listbox format. The various reports all use a similar format but with different headings and, of course, data. This program has several reports (or dumps) that include Dump Permissions for File System, Registry, Printers, Shares and Shared Directory. You can also “dump” Users, Groups, Policies, Rights and Services. These latter options are particularly useful for working directly with

Windows NT in areas not covered by TSM 3.6 (which allows us to manage the access to Files, Registry, Printers and Shares).

#### 3.2.1.2 Filemon

We used v4.0 of this GUI program, available from `http://www.sysinternals.com`. This layers itself above all the file systems in order to watch all file system activity. It can see all Input/Output Request Packets (IRPs) and FastIO requests that are directed at drives. The program can run continuously capturing all the activity on the system, or it can be started and stopped using a capture button.


File Monitor - Systems Internals: http://www.sysinternals.com
File
Events
Search
Drives
Help

```
#    Time         Process        Request                  Path                                           Result
263  12:02:38 PM  TASKMGR.EXE    IRP_MJ_READ*             C:\WINNT\system32\WIN32K.SYS                   SUCCESS
264  12:02:38 PM  EXPLORER.E...  FASTIO_QUERY_OPEN        C:\WINNT\System32\taskmgr.exe                  SUCCESS
265  12:02:39 PM  DISTASST.EXE   FASTIO_QUERY_OPEN        C:\DISTASST.PS                                 SUCCESS
266  12:02:39 PM  DISTASST.EXE   IRP_MJ_CREATE            D:\DISTASST.PS                                 FILE NO...
267  12:02:39 PM  DISTASST.EXE   IRP_MJ_CREATE            C:\acrobat3\distillr\In                        FILE NO...
268  12:02:39 PM  EXPLORER.E...  FASTIO_QUERY_OPEN        C:\WINNT\System32\taskmgr.exe                  SUCCESS
269  12:02:40 PM  EXPLORER.E...  FASTIO_QUERY_OPEN        C:\WINNT\System32\taskmgr.exe                  SUCCESS
270  12:02:41 PM  System         IRP_MJ_SET_INFORMATION   C:\WINNT\Profiles\Administrator\ntuser.dat.LOG  SUCCESS
271  12:02:41 PM  System         IRP_MJ_SET_INFORMATION   C:\WINNT\Profiles\Administrator\ntuser.dat.LOG  SUCCESS
272  12:02:41 PM  System         IRP_MJ_SET_INFORMATION   C:\WINNT\Profiles\Administrator\ntuser.dat.LOG  SUCCESS
273  12:02:41 PM  TASKMGR.EXE    IRP_MJ_CLEANUP           C:\WINNT\Profiles\Administrator\Desktop\       SUCCESS
274  12:02:41 PM  TASKMGR.EXE    IRP_MJ_CLOSE             C:\WINNT\Profiles\Administrator\Desktop\       SUCCESS
```

*Figure 11. Filemon - Sample Capture*

As shown in Figure 11, the information is displayed in columns that have the following meaning:

| | |
|---|---|
| **#** | Index number |
| **Time** | Time of request |
| **Process** | The process that executed the request. Examples shown include System, oserv.exe, csrss.exe, and explorer.exe. |
| **Request** | The request itself. In Figure 11 we can see requests such as IRP_MJ_Write, Close, Create, and FastIO_query |
| **Path** | Is the path of the resource, the file and/or directory. |
| **Result** | The result of the request such as Success, Access denied, File not found, and Buffer Overflow. |
| **Other** | Possible values include request, offset, and lengths, but we do not need to discuss them here. |

The use of this monitoring software was ideal for watching what resources were being used and what the results of the access were (such as success or access denied) and since it shows the processes and files in detail, there was no guessing as to which file was involved and what operation was denied or permitted.

Like most monitoring programs, we found it best to stop capture, clear the screen, and start capture just before executing a user activity to get the best results.

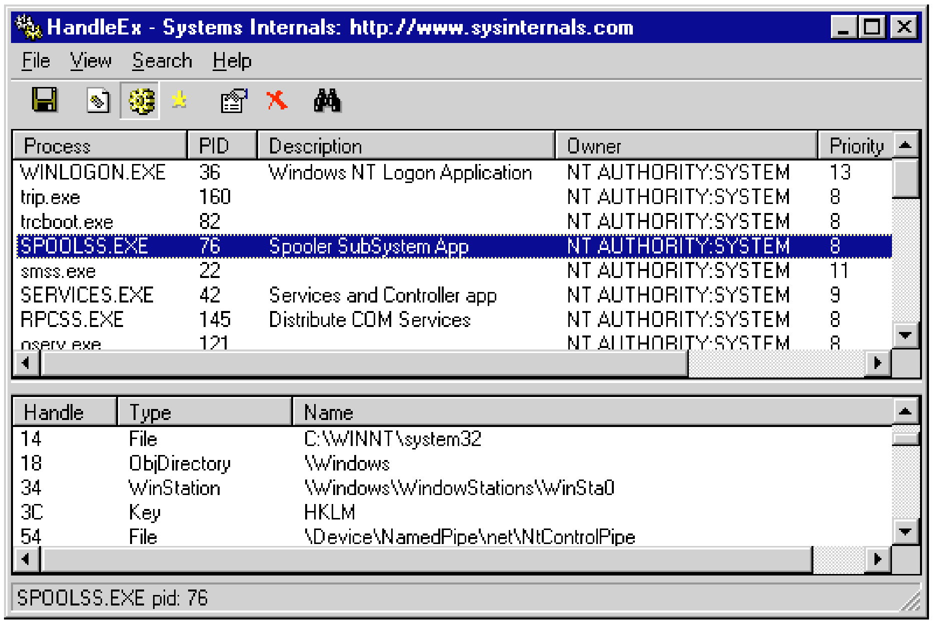
#### 3.2.1.3 HandleEx

The HandleEx program from `http://www.sysinternals.com` shows you, using a two-window GUI, what files, registry keys, and other objects have been opened by processes or which DLLs they have loaded. These are separate views; the DLL view is not discussed here, although if there are Windows errors on a DLL file, you can use this to see where the conflict is. Usually, an access conflict will come from locking a subdirectory since locking down individual DLL files was not found to be productive, and we recommend you do not do it. The most important feature of this software is the ability to show you who owns each process.

**Note**

The Windows NT architecture runs most, if not all, of its processes as SYSTEM owner. Access to any files and registries was not hindered by anything we used TSM 3.6 to lock down.

As shown in Figure 12, in the upper window of the program, you have a view of the process, such as `WINLOGON.EXE` or `SPOOLSS.EXE`, the process identifier (PID), description, owner of the process, a priority number, and the number of handles (not shown in the figure). We were not concerned with the priority or number of handles for the lockdown module. Although not used here, the DLL view can be obtained by clicking on the DLL button on the taskbar.



| Process | PID | Description | Owner | Priority |
|---|---|---|---|---|
| WINLOGON.EXE | 36 | Windows NT Logon Application | NT AUTHORITY:SYSTEM | 13 |
| trip.exe | 160 | | NT AUTHORITY:SYSTEM | 8 |
| trcboot.exe | 82 | | NT AUTHORITY:SYSTEM | 8 |
| SPOOLSS.EXE | 76 | Spooler SubSystem App | NT AUTHORITY:SYSTEM | 8 |
| smss.exe | 22 | | NT AUTHORITY:SYSTEM | 11 |
| SERVICES.EXE | 42 | Services and Controller app | NT AUTHORITY:SYSTEM | 9 |
| RPCSS.EXE | 145 | Distribute COM Services | NT AUTHORITY:SYSTEM | 8 |
| qserv.exe | 121 | | NT AUTHORITY:SYSTEM | 8 |

| Handle | Type | Name |
|---|---|---|
| 14 | File | C:\WINNT\system32 |
| 18 | ObjDirectory | \Windows |
| 34 | WinStation | \Windows\WindowStations\WinSta0 |
| 3C | Key | HKLM |
| 54 | File | \Device\NamedPipe\net\NtControlPipe |

SPOOLSS.EXE pid: 76

*Figure 12. Handlex - Sample Capture of System Processes*

The lower window shows the handle number in hexadecimal, the handle type which encompasses keys, files, threads, ObjDirectory, and WinStation. The last column shows the name of the handle, which, for some handles, is also a directory and file path.

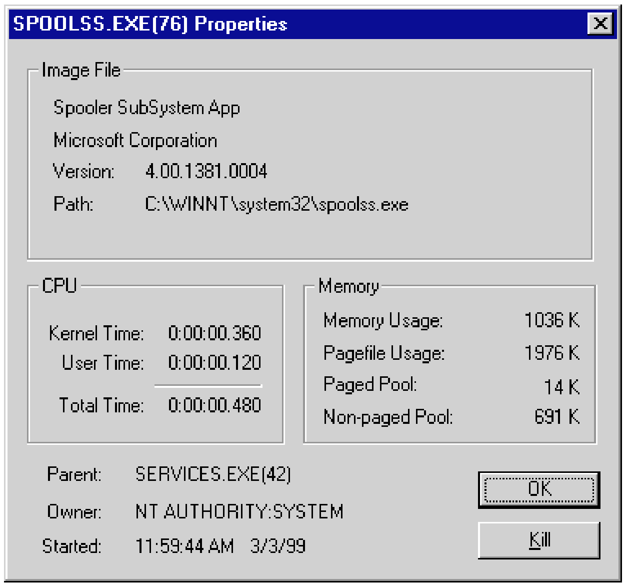

SPOOLSS.EXE(76) Properties

Image File

Spooler SubSystem App
Microsoft Corporation
Version: 4.00.1381.0004
Path: C:\WINNT\system32\spoolss.exe

CPU

Kernel Time: 0:00:00.360
User Time: 0:00:00.120
Total Time: 0:00:00.480

Memory

Memory Usage: 1036 K
Pagefile Usage: 1976 K
Paged Pool: 14 K
Non-paged Pool: 691 K

Parent: SERVICES.EXE(42)
Owner: NT AUTHORITY:SYSTEM
Started: 11:59:44 AM 3/3/99

*Figure 13. Processes Properties Box*

Right clicking on a process, such as SPOOLSS.EXE in Figure 12, brings up a second window, shown in Figure 13, giving the properties of the process. This includes the process path, owner, and whether it has a parent process. Note that care should be taken since this window gives the ability to kill the process in a similar way to the Windows NT Task Manager.

Refer to 4.6, “The Windows NT Operating System Lockdown Module” on page 83 to see the results of our use of these tools.

### 3.2.2 Watching UNIX with the TACF Trace

In the case of UNIX operating systems, the TACF security layer provides us with the necessary information about the TACF activities relating to applying rights to resource requests. The trace also gives us a good idea of which resources are being accessed by what process and therefore a good understanding of how an operating system or application works.

This again is helpful in identifying the resources we need to lock down. With TACF, we can also grant conditional rights for access through a certain program, and so, we can use the trace to check what conditional accesses we might want to grant in order to create an application Lockdown Module.

Conditional access is supported for the UNIX endpoint entity PROGRAM type. This specifies access to a resource when executing the specified program or application. For example, you might specify that updates to database files can only be made through the database application binaries. Note that there is no equivalent feature we can exploit on current versions of Windows NT.

#### 3.2.2.1 Controlling a TACF Trace

The TACF Trace is controlled through the TACF control console, which is invoked by:

secons `command [parameters]`

The secons utility provides a control console to the TACF daemons and performs operations such as:

- Controls tracing of the TACF authorization daemon (seosd)
- Enables and disables login
- Gets login status
- Displays run-time statistics
- Shuts down the TACF server daemons

A deeper discussion of all the secons commands can be found in the product documentation in the *Security Management Reference Manual for TACF*. In this chapter, we are only interested in the trace control commands that will be explained in more detail:

| | |
|---|---|
| -t+ | Enables tracing - causes the TACF daemon seosd to dump messages that specify its operations and actions to the trace file. |
| -t- | Disables tracing - stops the TACF daemon seosd from dumping messages to the trace file. |
| -tt | Toggles tracing status between enabled and disabled. |
| -ts | Displays the current tracing status. |
| -tc | Clears the trace file - removes all records from the trace file. |
| -tv [size in KB] | Online trace view - starts a browse session on the trace file and operates in a manner similar to the `tail -f` system utility. If the trace is running, you will see entries *live* as they are added to the trace file. Optionally, you can specify a size so that only the last |

| | |
|---|---|
| | kilobytes, as specified, are shown. To stop this operation, press the Ctrl-C key combination |
| -tv [-file <filename>] | Browse the specified file instead of /usr/seos/log/seosd.trace. This option can be used, whether seosd is running or not. |

For instance, you could use the following command sequence to run a TACF trace. Note, that the above mentioned control commands are only available to the TSM administration account (tmesec by default), therefore, unless we have added other TSM administration accounts, we have to log in as tmesec in order to run the trace.

1. secons -tc to clear the logfile
2. secons -t+ to enable tracing
3. run the target command, program or program action
4. secons -tt to toggle the trace status to disabled or
   secons -t- to disable the TACF trace completely

**Note**

Before starting the TACF trace, make sure there is sufficient space in the file system. The trace file can grow quickly, depending on the applications that are running and the resources that are locked.

Depending on the program to trace and the time the trace is running, the trace file should be cleared from time to time to guarantee a file size that still allows you to invoke an editor on the trace file (such as `vi`) for detailed analysis.

**Note**

After shutdown and restart of the seosd daemon the tracing is disabled by default, although the last active trace file is kept. This behavior as well as the name and type of the trace file can be changed by editing the seos.ini file. Refer to the product documentation for details.

#### 3.2.2.2 Reading a TACF Trace

We found it useful to have at least two windows open while tracing. One showing the complete tracefile, the other filtering for Warning Mode or Deny messages, depending on the default resource rights. This can easily be done by piping the trace file output to the grep command:

```
tail -f /usr/seos/log/seosd.trace | grep "Result: ’D’"
```

This will show events one might not have noticed while watching all the TACF messages at once.

The use of `tail -f` is recommended over the secons view option (`secons -v`) if the logfile /usr/seos/log/* (/usr/seos is linked to /usr/local/Tivoli/TACF) is defined as a Tivoli security resource itself. Otherwise, every request of the `secons -v` command will be individually validated and logged by TACF.

For further investigation, it might be useful to clear the trace file just before starting the target application or to send a message to the tracefile as a marker by invoking `secons -m <message>`.

```
09 Mar 1999 15:34:41> INET    : P=6694 , from 9.3.240.117:1351  port 23
09 Mar 1999 15:34:41> INET    > Result: 'P' 9.3.240.117->23, [0,-1], stg=408 gtsg=408
                Why?    Default access of TCP service
09 Mar 1999 15:34:41> FORK    : P=6694  U=0    G=-1   Child=14626
Pgm:/usr/sbin/inetd
09 Mar 1999 15:34:41> FILE    : P=14626 (/usr/sbin/inetd) U=0    (D=a0004    I=889   )
READ,  :/etc/passwd
09 Mar 1999 15:34:41> FILE    > P=14626 (/usr/sbin/inetd) U=0   /etc/passwd
BYPASS
09 Mar 1999 15:34:41> FILE    : P=14626 (/usr/sbin/inetd) U=0    (D=a0004    I=890   )
READ,  :/etc/security/passwd
09 Mar 1999 15:34:41> FILE    > (/usr/sbin/inetd) Result: 'P' [stage=57 gstag=57
ACEEH=3    rv=0(/etc/security/passwd)]
                Why?    Resource ACL check for user groups
09 Mar 1999 15:34:41> EXECsu  : P=14626 U=0    G=0    (D=a0005    I=8543  )
Pgm:/usr/sbin/telnetd Attached to: 9.3.240.117
09 Mar 1999 15:34:41> EXECARGS: 'telnetd'
09 Mar 1999 15:34:41> EXEC    > Result: 'P' [stage=59 gstag=59 ACEEH=3
rv=0(/usr/sbin/telnetd)]
                Why?    Resource universal access check
09 Mar 1999 15:34:41> FILE    : P=14626 (/usr/sbin/telnetd) U=0    (D=a0004    I=817
)    READ,  :/etc/netsvc.conf
09 Mar 1999 15:34:41> FILE    > (/usr/sbin/telnetd) Result: 'P' [stage=57 gstag=57
ACEEH=3    rv=0(/etc/netsvc.conf)]
                Why?    Resource ACL check for user groups
09 Mar 1999 15:34:41> FILE    : P=14626 (/usr/sbin/telnetd) U=0    (D=a0004    I=133
)    READ,  :/etc/hosts
09 Mar 1999 15:34:41> FILE    > (/usr/sbin/telnetd) Result: 'P' [stage=57 gstag=57
ACEEH=3    rv=0(/etc/hosts)]
                Why?    Resource ACL check for user groups
```

*Figure 14. Sample TACF Trace - The Beginning of a Telnet Session*

Figure 14 shows us a sample TACF trace output. It was taken while initiating a telnet session to a locked-down system.

The TACF trace messages begin with a date and time prefix ending in the greater-than sign (>) followed by an event type word in uppercase letters (such as INET or FILE) and a symbol such as a colon (:), exclamation point (!), or the greater-than symbol (>).

The meanings of these symbols are shown in Table 1:

*Table 1. TACF Trace Symbols*

| Symbol | Meaning |
|---|---|
| colon (:) | TACF was signaled for an event or took an action. |
| greater-than (>) | TACF made an authorization decision resulting in **D** (Deny), **P** (Permit) or **BYPASS**.<br>BYPASS indicates that the event did not require the interpretation of an access rule - For example, a setuid request to the same user ID as the current user ID. |
| exclamation (!) | TACF detected an error - for example a request from an unknown process. |

There are at least 35 events with many more subevents. The event arguments are explained in detail in the S*ecurity Management Reference Manual for TACF,* which is product documentation. Fortunately, one can easily guess what most of them mean because of the descriptions given in the trace. TACF will generally tell you what request took place (that is what was intercepted) and the next entry will usually be the access decision made by TACF. So, in the first two entries in the example in Figure 14, we can see that TACF intercepted an incoming internet request. This request detail includes the IP address of the issuer `9.3.240.117` from local port `1351` and the port number (23 - telnet) that was requested. The second entry shows the access result (explained next in the FILE event example).

In the first FILE event, we can see some additional important information you can get out of the trace file. There is a `U=0` entry. This is the user ID under which a program is running. This is significant because TACF applies its rights to the original user ID. For example, if a user logs in under their own name and then uses the `su` command to switch to the root user, TACF still regards them as the original user. This is a useful feature for delegating the authority of administrators. You can allow someone to `su` to root so that UNIX will allow a function to be performed, but you can still use TACF to restrict what the user can do based on the ID they logged in with.

Near the end of the example, in Figure 14, for instance, the telnet Daemon is running with root permissions. This can be seen by observing the U (`U=0`) argument.

**Problem Determination Tip**

By default, TACF maintains a *look-aside* buffer database of UNIX user IDs. If this database goes out of synchronization with the system files, TACF may think that a user is someone other than whoever they logged in as resulting in incorrect access checking. The trace will show you who TACF thinks is performing the request and you can match this back to login names with look-aside database commands. Refer to the `sebuildla` command in the product documentation for more details.

The FILE event type also has a request log entry and a result log entry. In the first FILE event, we can see a request from process 6694 (P=6694) /usr/sbin/inetd associated with user ID 0 (U=0) for READ access to /etc/passwd. If you really want to get into the details, the trace entry also shows the device and i-node of the file being accessed (d=a0004 I=889).

After the TACF authorization decision is made, it is stated in the trace in an entry where a greater-than symbol follows the event type. You will see something like `Result: 'P'` at the beginning of the entry or `BYPASS` at the end. P indicates permit, a D would indicate Deny and BYPASS means TACF did not need to invoke a rule. Knowing who accessed what resource and in what manner is the information necessary to create adequate resource definitions in our lockdown module, as well as default and role-based resource access rights.

The result of a deny or permit also includes a *Why?* statement showing the reason why a deny or permit resulted. The result of this Internet request as shown in line two is `'P'` (permit). The detailed reason text including the `Why?` keyword is the description that indicates which stage and granting stage phase of the decision flow made the final decision to permit or deny the TCP/IP service for the requesting host. In our case (stage 57), the `Default access of TCP service` granted permission. This correlates to the default access specified in the resource record in our security profile for the UNIX:TCP resource named telnet, or, if there is no resource for telnet defined. Stage and granting stage codes are in the TACF product documentation in the TACF Status Codes chapter.

**Note**

TCP Services must be specified as a name if the name is defined in the /etc/services file, otherwise you must specify the service by port number.

In the next line, the TACF trace reports that the above-mentioned process 6684 running with user ID U=0 (root permission) forked (FORK:) a child process with the number 14626. The program running in the parent process (and initially also in the child process) is the internet daemon inetd, shown with the full path Pgm:/usr/sbin/inetd. Note that TACF never denies a fork request; it is always granted.

The next three lines indicate that our inetd process attempted to access a TACF-protected file stated by the event argument FILE:. As mentioned previously, the trace shows the device (D=a004) and inode (I=889) of the requested file as well as the access mode (READ) and the accessed file (/etc/passwd).

The keyword BYPASS indicates that the event did not require the interpretation of an access rule. Compare this to the next event line where the inetd process tries to access /etc/security/passwd in READ mode and a decision was made with the Result: 'P', indicating a permit result. The stage and granting stage (gstage) are mapped to a text string reason. In our case, the Resource ACL check for user groups granted permission. This equates to permissions granted to a role in a security profile.

After checking the UNIX password files, the Internet daemon invokes the telnet daemon. Because the program /usr/sbin/telnetd is a setuid or setgid program, the EXECsu: event argument indicates the invocation under user ID 0. TACF will decide whether to grant its execution by invoking the database access rule decision mechanism. If the ip-address to which the process is attached is extractable, TACF reports this in the message text (Attached to: 9.3.240.117).

As the sample TACF trace shows, the telnet daemon itself accesses two more files with the Result: 'P' (permit) in both cases.

## 3.3 Resource Considerations

This section looks at specific considerations for using TSM system policies and for looking at resources when protecting operating systems or applications. By discussing examples of resources to protect, the aim is to help you determine what would apply in your environment.

We look at these topics grouped in the following way:

- System Policies
- Windows NT Considerations by resource type

- UNIX Considerations by resource type
- Application considerations

You will notice that almost all of the sample LDF files are for UNIX applications. This is because we wanted to include examples of conditional access and other resource protection such as TCP services. For some applications, the Windows NT LDF will be something of a subset of the UNIX equivalent. The path names will, of course, need alteration, but the majority of the work regarding pieces to consider has been done in the UNIX LDF.

## 3.3.1 System Policy Considerations

System policies apply to the whole target operating system, not something that is specific to individual users, groups or resources. This means that every user of the system must follow the policy or rules. Although TSM for Windows NT and TSM for UNIX (with TACF) are slightly different in their management implementation, they both use the same system policy record. Windows NT implements different policy parameters than UNIX. For example, A UNIX system policy can manage a policy to use only alphanumeric characters in a password. This will be explained in more detail later in this section, but Windows NT does not have this ability. Attributes defined in the system policy record that are not applicable to the endpoint type are ignored during profile distribution. To show what parameters they do share, we can use the policy parameter for password length as an example. If the system policy requires a password of 9 characters or more, this would apply to both a UNIX user and a Windows NT user.

Since system policy is not something that needs to be used in every security profile, we have not used it in the LDFs. We have included more detail on system policy in this section for background information to help you determine adequate system policies.

System policy consists of a Password Policy, Login Policy and for TACF there is a Resource Type Access Policy. In TSM 3.6.1 we have the added capability of specifying audit policy for NT system events.

### 3.3.1.1 System Password Policy

Granting access to computer systems through an authentication process of user name and password is universal. The password policy defines requirements and regulations in the use and makeup of these passwords. This can ensure strong passwords and gives fewer chances for passwords to be compromised which, in turn, leads to fewer intrusions. The password policy regulations and some suggested parameters are as follows. For a further explanation of these, see the redbook *Tivoli Security Management*

*Design Guide*, SG24-5101, or the *Tivoli Security Management User's Guide* which is product documentation.

***Maximum Days between Password Changes***
The maximum number of days between Password Changes is 45 to 60 days; this depends on the production site. If the amount of users is great and there is no *single sign-on* solution, users may have to remember several passwords. Then, you may want to use the extended time of 60 days. More than 60 increases the window of opportunity if a password has been compromised.

***Minimum Days between Password Changes***
The minimum number of days between password changes is 7 days; use this minimum password age value in conjunction with *Password history depth* to stop users from cycling between two favorite passwords.

***Password History depth***
The password history depth is 10 passwords; if we look at the math, it gives us 70 days before the user can revert back to a previously used password.
*min. password age x password history depth = min. password reuse time*

***Minimum password length***
The minimum password length is 8 to 9 characters; The Windows NT password length should be set over 7 in case any LANMAN hashes are used; this makes it much harder to break the password. Passwords over 9 will be too long and cause the user to write down the password in most cases. LANMAN hashes are used in environments where LANMAN authentication is allowed alongside Windows NT authentication (which is the default). This allows older Windows and other servers to participate in a Windows NT domain. Refer to Microsoft Knowledge Base article Q147706 for more information about LANMAN authentication and how to disable it.

Allowable character requirement parameters in TSM for Unix only are as follows:

***Minimum Alphabetic***
One alphabetic character must be included.

***Minimum Numeric***
One numeric character must be included.

***Minimum Alphanumeric***
One alphanumeric character must be included.

By putting 1 for each and in combination with 8-9 minimum password length, this assures an alphanumeric password of 8-9 characters long, which is what we suggest since most password cracking programs do not handle a mix of alphabetic and numeric very well. The inclusion of special characters is even more desirable as shown later.

***Minimum Uppercase***
Minimum Uppercase is zero; we recommend this not be a requirement since case may not be significant in some platforms in a multi-platform environment.

***Minimum Lowercase***
Minimum Lowercase is zero for the same reason as above.

***Maximum Repeated***
Maximum Repeated is three; this gives the user the ability to use a character up to 3 times consecutively in a password.

***Minimum Special (Special Characters - !,*,?)***
Minimum Special is zero to one; Zero for normal production environments and one for more sensitive areas since inclusion of special characters makes the password much harder to *guess*.

**UNIX Password Setting**

Password quality checking for TACF depends on the `setoptions` configuration and the user must use sepass to make password changes in order to have the quality checks take place.

#### 3.3.1.2 System Login Policy
This policy deals with the regulations on actually logging into a system. Consider it as the next logical step from the password. Once password rules have been accepted or followed, there are rules for the actual login.

***Account Lockout***
Lockout duration should be 60 minutes. Much less and users with malicious intent would have less time to wait to continue the *attack* and if it is set any higher, it will stop productivity for a user who may just be having a bad day with typing.

**Note**

The root user in Unix and the Administrator user on Windows NT (if not renamed or assigned to another user, it would apply to them) cannot be locked out. Note also that Sun OS 4 does not support lockouts.

***Failed Login Attempts***
It is not recommended that a value less than 3 be used. A user needs to be given a couple chances to reenter a password due to possible typing errors.

***Time span setting for lockouts***
Here, NT and UNIX implement this in different ways, but a happy medium can be found based on the customer requirements. UNIX, or, in this instance, TACF, considers the time span as the whole period for counting failed logins, and for Windows NT, time span is how long between each login to wait before resetting the count. For Unix, the setting should be 60 minutes, and for Windows NT, it should be 30 minutes. In determining a happy medium, just remember that the larger the value, the fewer chances a hacker has to try a different password each day. However, at the same time, the larger the value, the greater chance there is of a genuine user mistyping passwords three times (or whatever the value) within the resultant time period.

#### 3.3.1.3 System Resource Type Access Policy
We do not implement TACF Resource Type Access Policy in our lockdown modules. This policy allows you to specify a default for an entire resource type if nothing else is specified. This may be required in very sensitive environments, but, depending on the number of accesses, this can have a significant performance impact when every access to a resource of the set type must be checked. In fact, even if you set this for the UX:FILE resource type, this will have no effect. TACF ignores this due to the potential performance impact.

### 3.3.2 Windows NT 4.0 Considerations
In order to decide which resources and access control levels we needed to implement in our lockdown module with TSM 3.6, we found that an approach based on an understanding of the resource types and their limitations and strengths was the best course of action. Several authoritative documents were then reviewed which gave more precise security resource lockdown suggestions. One document we read was *Windows NT Security Guidelines - A study for NSA (National Security Agency) Research* by Trusted Systems Services. This document looked at Windows NT’s C2 compliance and broke down the NSA *Orange Book* to describe what methods and resources must

be taken to closely match that of the C2 compliance. The second document was a paper from Microsoft, *Securing Windows NT Installation*, October 23, 1997, Microsoft Corporation. This document was mostly helpful from the registry perspective.

We will now look at the resource types we can manage through TSM.

**Note**

TSM 3.6.1 provides enhanced capability when managing Windows NT resource types. Our work was focused around 3.6, and where applicable in the following sections, we mention what else could be considered using TSM 3.6.1.

In many implementations, we expect that Tivoli administration will be used for setting global policies in areas that can apply to multiple targets; local Windows NT administrators will continue to use Windows NT administration tools to perform certain *fine-tuning* functions. The aim should be to avoid using local tools to perform administration functions that are also performed using TSM - this will prevent synchronization problems.

#### 3.3.2.1 NT:DIRECTORY

This resource type protects accesses to NTFS directories. The directory access ACL is modified by an NT:DIRECTORY record. Every NTFS file and directory has an owner who has a special status. The owner of that resource can manage the permissions of that resource in the same way as the Administrator account. They can even deny the Administrator the right to access that resource. So resources created by the user are owned and managed by the user unless the administrator takes ownership away. As of TSM 3.6 and 3.6.1, TSM does not manage the way Windows NT treats the permissions of file owners (specified through the Windows NT group CREATOR OWNER). If changes are required, they must be done using native Windows NT tools.

#### 3.3.2.2 NT:FILE

The NT:FILE resource type sets restrictions on the ability to perform actions on files in an NTFS file system. Windows NT's native security structure consists of two permission ACL's in the file/directory resource: Directory ACLs and newly created file ACLs, shown in native NT as two sets of permissions after the directory name as in C:\Directory (RWX) (RWX).

The implementation of asterisk wildcard characters (*) for directory and file resources as in C:\WINNT\System32\* changes between 3.6 and 3.6.1. In

3.6, a file resource specified with an asterisk only sets the default permissions for all files subsequently created in the specified directory after the profile is distributed, not for any files already in the directory. No other wildcard characters such as question marks are allowed.

In 3.6.1, we can use the question mark wildcard character (?) and we have the option to specify existing files, subsequently created files, or both. Our lockdown used the 3.6 model.

In 3.6, if you wished to apply permissions to existing files, you either needed to explicitly specify each file (not recommended due to the amount of resource records that would be required) or use the Windows NT utilities to “Replace Permissions on Existing Files or Subdirectories” which then pushes the permissions to existing files and/or subdirectories. (Note that TSM 3.6.1 has added the capability of replacing permission on existing files and subdirectories).

Even in securing applications, as already stated, we suggest limiting the use of security applied directly to individual files with the security NT:FILE resource only for important single files for the following reasons:

- It is too much work to administer every file separately. You would have to define one security profile record of the type NT:FILE for each file.
- When you define access permissions directly for a single file, they will be set when the Tivoli profile is distributed to the target (managed node or endpoint). If you locally set the file permissions of the directory in which the file resides, the previous set of file access permissions can be overridden because Windows NT provides an option to apply new directory permissions to all files already in the directory. Using the TSM 3.6.1 **Apply Permissions** option can also override permissions set on individual files depending on the order in which profiles are distributed.

Therefore, if you are setting access permissions on single files, you have to make sure that the default permissions of the directory in which the files reside are not modified locally on the Windows NT system since they could potentially be applied to all the files in the directory overwriting permissions set on individual files.

The TSM 3.6.1 improvements are implemented in the Resource Default Access panel and Resource Access Audit Control panel. From the CLI, the new parameters include two additional key words: Apply and Recurse.

| | |
|---|---|
| Apply Existing | Allows permission values to be applied to existing files that match the wildcard pattern. |

| | |
|---|---|
| Apply NewOnly | Allows permission values to be applied to only newly created files. (3.6 capability). |
| Apply All | Allows permission values to be applied to both newly created and existing files. |
| Recurse | Allows permission values to be applied to all subdirectories (not just those matching the wildcard) similar to the /S flag on the `NT DIR` command. |

#### 3.3.2.3 NT:SHARE

This resource type restricts access (through the network) to shared directories and their contents. This lockdown file does not include share resources due to the site-specific nature of these resources (See also 4.6, “The Windows NT Operating System Lockdown Module” on page 83). Native administration Windows NT shares that contain the dollar sign ($) are not managed by TSM 3.6. Note that auditing of shares is not supported in NT; so, it is also not supported in TSM 3.6. It is a common practice to audit the files and directories themselves on the host machine. You are not auditing the share, but the file and directory as a resource.

If you want to place different access restrictions on individual files or subdirectories that are available on the network in a share, you must create separate resource records for these files and subdirectories. In other words, the share of share\sub\file1.dat with No Access is one resource and share of share\sub\file2.dat with Read/Write is another resource. Share resources cannot use wildcards, they must be name-specific. TSM 3.6.1 revised the ability of the TSM Populate tool to allow itself to populate resource type NT:SHARE using either the GUI or `wpopulate` / `wpopsec`. TSM 3.6 would only populate the SYSTEM (user rights) resource in addition to the Roles and Groups, of course.

#### 3.3.2.4 NT:REGISTRY

This type sets restrictions on the ability to edit and control entries in the NT Registry. The first thing to note is that it is not possible to use wildcard characters to specify multiple registry hives or keys. There are several means within Windows NT to access and edit the registry. For example, you can use the utilities in the Windows NT Control Panel or Setup Applications. Here, concerns with securing applications appear.

By using the Filemon or HandleEX programs mentioned in 3.2.1, “Watching Windows NT Applications” on page 32, we watched the programs as they executed and installed to see which registry hives and keys they required. We have to be able to protect the registry without preventing applications from installing for example (unless that is our aim). One way of protecting the registry is by preventing the use of applications that modify it. There is the possibility to change values of the registry directly using the operating system with `REGEDIT.exe` or `REGEDT32.exe`. In our lockdown module that you can see in 4.6, "The Windows NT Operating System Lockdown Module" on page 83, has denied access to the `Run...` command, `CMD.exe` and various other executables including `REGEDIT` and `REGEDT32`. There are many shareware tools that make changes to the registry. Therefore, it is of little use to protect the registry by restricting the access to the Windows NT registry editor. Users can find other ways to make modifications. Unless you directly manipulate the capabilities of the CREATOR OWNER group, users can pull shareware tools from the Internet, install them, and have full control of the programs.

Our Windows NT lockdown example has locked out users from installing to a degree, but the possibility is always there. The Windows NT registry has to be secured with its native security. That is, we must create NT:REGISTRY resources to modify registry ACLs to prevent unauthorized changes. This is a topic that is also covered in the redbook *Tivoli Security Management Design Guide*, SG24-5101.

The default permissions of the hives are set reasonably well in NT 4.0; with some exceptions, the Everyone group has read access to all the data. In our lockdown file, we have used the Microsoft recommendations from their paper *Securing Windows NT Installation* in locking down access to hives and keys from the Everyone group (default access=No Access). Care should be taken not to modify the access for the SYSTEM group. The operating system needs to have access to all the information.

#### 3.3.2.5 NT:SYSTEM

The NT:SYSTEM resource defines the user rights for a Windows NT system. This resource type can be given any name, but only one SYSTEM resource should be distributed to a machine. There is no auditing of this type. We found it most useful to simply populate user rights from a default Windows NT installation. Populating creates the SYSTEM record named User Rights.

The *Tivoli Security Management User's Guide* product documentation lists all the rights and their descriptions in the NT Resource Types appendix. Examples of rights include Shutdown, Remote Access, and Install Devices.

In our lockdown, this resource was the key in security role development. In order to make the basic lockdown flexible for the different production sites, very few if any NT:FILE or NT:DIRECTORY resources were named for the various security roles. The NT native User Rights were sufficient with an underlying restrictive security group (such as Domain Users). This is explained in 4.6, “The Windows NT Operating System Lockdown Module” on page 83. Key items to note about our lockdown with this resource type is that Local Logon and Remote Access were allowed for everyone due to the domain structures that again, differ with different sites. Backup, a right given to Backup operators, overrides access permissions granted to individual files and directories. The TSM User Rights properties closely map to those of native Windows NT.

#### 3.3.2.6 NT:PRINTER

The NT:PRINTER type represents a logical printer defined within the Windows NT Print Manager. Each PRINTER record sets restrictions on the ability to submit jobs to a printer and control jobs sent by others. This lockdown file does not include PRINTER resources due to the site specific nature of these resources, but mention will be made of them in 4.6, “The Windows NT Operating System Lockdown Module” on page 83. If you intend to implement printer resources in TSM 3.6, you should be aware that there is no Manage Documents permissions for this resource type. only Access - the ability to print, No Access, and Full Control - the ability to print and manage documents. TSM 3.6.1 has added the management of the Windows NT permission of Manage Documents. It also has the added capability to populate NT:PRINTER by using the TSM Populate tool in the GUI or wpopulate / wpopsec.

### 3.3.3 UNIX Considerations

The Tivoli Access Control Facility (TACF) is an object-oriented security system. The TACF database describes two types of objects: accessors and resources. Users and groups are accessors: they access resources. Resources are objects to be protected, such as files and services. Each record in the TACF database describes either an accessor or a resource.

Each object belongs to a class. A class is a collection of objects of the same type. For example, TERMINAL is a class containing objects that are terminals protected by TACF. The concept of classes or types of resources is well known to mainframe security administrators.

In this section, we will describe TACF resources and classes discussing considerations to take account of when using them to lock down UNIX or an application running under UNIX.

#### 3.3.3.1 TACF Resources

In TACF terminology, a resource is an entity that can be accessed by accessors (users and groups). The most common type of resource is a file. You access a file when you read information from it or write information to it.

Other types of resources are terminals (accessed when you sign on) or directories.

The properties of the protected resource are stored in the resource’s record. A record is a collection of data consisting of the name and properties of a resource or accessor. These properties tell who defined the record, the date when the record was defined, and so on. In general, the most important information contained in a resource record is a list of the accessors that are authorized to access the resource. This list is referred to as the access control list (ACL).

A security resource may appear more than once in the list of resources assigned to a security role where each entry has unique access rights. For example, a security group may have different access to a resource when accessing it through a specific program (known as conditional access) than when accessing it directly.

**Note**

Resources within Role Resource Definitions can only be deleted if no conditional right access is defined.

#### 3.3.3.2 TACF Classes

A class is a group of object records that are of a similar type. For example, the TERMINAL class contains all objects that are of type terminal, such as tty1, tty2, and tty3. The FILE class contains definitions for files and file masks; the PROGRAM class contains the records that protect trusted programs from being modified.

TACF contains four types of pre-defined classes:

| | |
|---|---|
| Accessor | Objects that access resources |
| Definition | Objects that define security entities, such as security labels and categories |
| Installation | Objects that are protected by access rules |
| Resource | Objects that are protected by access rules |

Because our intention in this Redbook is to develop lockdown definition files in order to secure applications with Tivoli Security Management, we will concentrate on the *Resource* type classes. You can refer to the TSM product manuals for TACF for more information on the other classes and types. If you wish to make use of those classes, you can do so using the TACF command line interface (selang).

Most classes implemented in the current Tivoli Security Management Version 3.6 and accessible in Security Profile Resource Records are of the class type Resource. Exceptions include the UX:SECFILE which is the TACF class type of *Definition*. Furthermore, one can define UX:GROUP records through Security Group Records of type *Definition* as well as UX:USER records of type *Accessor*.

Note that there are currently 21 classes predefined by TACF, but the following classes are the ones currently managed through Tivoli Security Management. The list contains the class name in bold type followed by a description.

| | |
|---|---|
| **UX:CONNECT** | The CONNECT class protects the outgoing connection. Each object in this class defines which users can access which internet hosts. |
| **UX:FILE** | Each object of this class defines the access allowed to a file or directory. |
| **UX:PROCESS** | Each object of this class defines an executable file that, when running as a process, is to be protected against KILL attempts.<br>Major daemons and database servers are good candidates for such protection because these processes are the main targets for service-denial attacks.<br>Before defining the program in the PROCESS class, the file must also be defined in the FILE class. |
| **UX:PROGRAM** | This class defines programs that are considered part of the *trusted computing base*. Programs in this class are trusted programs; they cannot have security breaches and are monitored by the TACF watchdog daemon to ensure they are not modified. A system or security administrator can only define programs marked as setuid or setgid in the UNIX file system as trusted programs within TACF. |
| **UX:SECFILE** | The SECFILE class is similar to the PROGRAM class in that it stores information about program files. However, objects of the SECFILE class cannot appear in a conditional access control list. This class is intended to provide verification for important files in the system. The watchdog daemon scans these files and ensures the information known about these files is not modified. |
| **UX:SURROGATE** | This class defines restrictions that protect the user from other users when they make substitute user ID (su) requests. TACF treats the surrogate request as an |

abstract resource that can be accessed only by authorized users.

| | |
|---|---|
| **UX:TERMINAL** | The TERMINAL class defines objects that represent the terminals of the local host, another host on the network, or X-Terminals from which a login session can be made. Terminals are checked during user login. Users can sign on from a terminal only if they have been authorized to use the terminal. |
| **UX:TCP** | The TCP class includes records for TCP/IP services, such as mail, ftp or http. Each record's ACL can specify access types not only for individual hosts that may request the service, but also for HOSTNET and HOSTNP resources. Services must be specified as a name if the name is defined in the /etc/services file, otherwise you must specify services by port number. |

**Note**

The following three host classes are only accessible using the UX:TCP class - while defining TCP resource access rights.

| | |
|---|---|
| **UX:HOST** | Each object of this class defines a host (client). The host is identified by either its name or its IP address. For each client (HOST object), there is a property that lists the service rules that govern the services the local host may provide to the client while using Internet communication, for example, <hostname> or fully qualified <IP-address>. |
| **UX:HOSTNET** | Each object in this class defines a group consisting of all hosts on a particular network. HOSTNET objects define access rules that govern the access other hosts on the specific network have to the local host when they are using Internet communication. The name of each object consists of mask and match values for the IP address such as <Network IP-address Mask>/<Match Mask>. |
| **UX:HOSTNP** | Each object in this class defines a group of hosts where the hosts belonging to the group all have the same name pattern. Every HOSTNP object's name contains a wildcard, for example, <*.domain-name>. |

### 3.3.4 Application Considerations

By discussing each of the resource types in the previous two sections, we hope you have developed some ideas about what is reasonable to achieve in securing applications. In addition, 3.2.1, “Watching Windows NT Applications” on page 32 and 3.2.2, “Watching UNIX with the TACF Trace” on page 37 should help you start looking into what you need to do.

The lists in the next sections summarize things to consider when designing lockdown modules.

#### 3.3.4.1 Windows NT

The following is a list of considerations to keep in mind when protecting Windows NT installations with TSM:

- Most system processes run under Administrator rights.
- Administrator can’t be locked out using Windows NT features exposed in TSM. The administrator can be prevented from accessing a resource, but the administrator can also grant himself the rights to a resource. Protection of this type would involve restricting who has access to the Administrator ID and, instead, defining more users with the rights they need to perform administrative functions. Auditing should be used to monitor the actions of the true Administrator.
- While it would be nice to have some of the TACF resource types such as TCP and CONNECT, they are not currently available in Windows NT. Tivoli plans to implement the management of these capabilities when they are a part of the operating system, perhaps in Windows 2000.

#### 3.3.4.2 UNIX

In our experiments with lockdown modules, we felt it was worth documenting some of the considerations and differences between different flavors. Our LDFs are for Solaris and AIX; so, the notes came from our experiences with those, but similar considerations exist on all platforms:

- The AIX System Backup Facility needs to read all Files through some dedicated programs
- AIX uses the System Resource Controller to control certain Daemons.
- Part of your security policy might include the use of the /etc/securetcpip script on AIX which disables all non-secure daemons like rlogind and tftpd.
- Increasing the security of the hardware and operating systems (for example for use as Firewalls, DMZ-Servers, and so on)
  - Security in general

  - Deactivation of unneeded Services/Daemons
  - Penetration Tests
  - Password Policies
  - Backdoor Checks (setuid, setgid files, unowned files and so on)
  - Trusted Hosts

#### 3.3.4.3 Database Server Applications

Database servers handle some security features such as user authentication, but most may also require some access protection to avoid accidental or deliberate damage.

When looking at this type of application, consider the following:

- Does the application allow you to specify conditional resource access for only some programs or processes (mostly only the daemons)?
- Does the application usually employ a well structured file system or directory structure?
- Does the application keep database files, logfiles, and config files in separate directories (which makes it easier to design resource roles)?

#### 3.3.4.4 Application Server

Our experiences with Netscape Enterprise Server gave a very different example of application lockdown compared to the database servers.

- Few (one to four) daemon processes.
- Very complicated conditional resource accesses are required due to the use of a program for every little task.
- Directories tend to be less well structured from a TSM implementation perspective (such as configuration files and binaries being together in one directory.

# Chapter 4. Lockdown Descriptions

This chapter provides detailed descriptions of lockdown modules that we developed as examples. You will notice we have a heavy bias in the UNIX arena. There are a number of reasons for this. For example, the UNIX platform can be more challenging to address since we can handle conditional access of resources. What we can manage in terms of resources in Windows NT is mostly a subset of what we handle for UNIX. For many applications, we can take a UNIX application (LDF) and change it relatively easily to a Windows NT one by removing some resource types and changing file references to the Windows format.

The examples provided on the CD are for the following environments:

- A typical SAP/R3 implementation
- The Tivoli Framework 3.6 on UNIX
- Lotus Domino 4.6 on UNIX
- Netscape Enterprise Server 3.6 on UNIX
- Oracle database on UNIX
- The Windows NT Server 4.0 operating system
- IBM AIX 4.2 operating system
- Solaris operating system

See 1.1, “Obtaining Lockdown Updates” on page 5 for information about updates to the modules provided with this book.

**Note**

Although a Solaris LDF is included on the accompanying CD, no description appears for it in this publication. You can find more background information in the (more comprehensive) AIX module in 4.7, “The AIX Lockdown Module” on page 93.

These modules provide protection against unauthorized access. The features of TSM allow us to concentrate on file systems and, where appropriate, network access. One aim in the UNIX environment was to enable the protection of resources from accidental or other modification by the root user where this was practical.

During our investigations, we considered other applications suitable for lockdown modules. These might include office suites, mail servers, and other databases and operating systems.

When we considered things like Firewall products, we felt that these were in a similar category to the Tivoli Framework in terms of how applicable TSM might be (see 4.2, “The Tivoli Framework Lockdown Module” on page 62). Firewalls have special considerations, and the addition of TACF to a firewall system may only be required in certain special circumstances. A firewall already has many security features (and many of these can be managed by Tivoli in other ways as with CheckPoint Firewall’s integration with Tivoli and Tivoli User Administration). A firewall is likely to be physically very secure and very few people would be given root access to the machine. Most firewall vendors would prefer the system used for the firewall to be as *clean* as possible, that is, with all non-essential features removed or disabled.

The LDF contains all the definitions required to build a security profile including roles, resources, and access rights.

**Note**

In the LDF files we typically define one or more security groups. It is highly recommended that you actually apply the roles we define to your own group structure. An ideal Tivoli security group structure results in users being members of very few groups (maybe just one or two) to make user access control management simpler.

## 4.1 The SAP/R3 Lockdown Module

Extending the security of an SAP/R3 implementation will be very dependent on the implementation. The R/3 environment will most likely exist across multiple environments supported by TSM and will involve other applications for which lockdown modules may or may not already exist. The intention is that this LDF will provide a starting point on UNIX and that the review of other modules, such as “The Oracle Lockdown Module” on page 79, will help identify what changes may be necessary in any environment. The directory structure will at least be consistent across UNIX variants. Note that with Windows NT, there will be multiple usr/sap/SID directories on multiple disks that will need to be maintained.

Note that this LDF has a number of variables that will need to be identified and defined prior to implementing it including the three character system identifier (SID) for the R/3 system and the database.

### 4.1.1 TSM Roles in the SAP/R3 LDF

We defined two roles in the LDF with sample groups as follows:

```
GROUPS
  SAPAdmin "SAP Administrator" =  (${SID}adm) audit(LF,RF) ;
  SAPSystem "SAP System" =   audit(LF,RF) ;
ROLES
  "SAP System" "SAP System" = (SAPSystem) ;
  "SAP Administrator Role" "SAP Administrator Role" = (SAPAdmin) ;
```

The roles were determined as follows:

**SAP System**
This role is for the R/3 administrators typically defined in the sapsys UNIX group. It contains the DVEBMGS resources with Read and Execute access. We also specify All access to the transaction files and directories and the home directory. In our sample groups, we gave this role to the SAP System group which should be populated with members of the UNIX sapsys group.

**SAP Administrator Role**
This role is for the owner of all the R3 processes on the system as well as the file system directories and files that are used to run the system. This contains all the resources of the LDF with all access to those resources. In our sample groups, we gave this role to the SAP Administrator group and it is likely that this will contain very few - possibly just one user ID. This is the ID SAP will use to start/stop the R/3 system, administer the file systems, and so on.

### 4.1.2 TSM Resources in the SAP/R3 LDF

The directory structure revolves around the system identifier (SID); so, in the LDF, we use a variable ${SID} to identify this to the resource declarations. You will need to specify this variable.

The SAP binaries and other installed data are stored under the /sapmnt directory structure, and we defined default access of Read and Execute to these directories with the exception of /sapmnt/${SID}/global where we specified No Access as the default. These file systems do not need to be accessed outside of the roles we specify.

```
RESOURCES
  /sapmnt/${SID}/exe  = [RX] audit(RN)  ;
  /sapmnt/${SID}/global  = [N] audit(RN)  ;
  /sapmnt/${SID}/gui  = [RX] audit(RN)  ;
  /sapmnt/${SID}/profile  = [RX] audit(RN)  ;
```

In the DVEBMGS directories, we specify the SAP instance through a variable. You must set this variable for the LDF. You could add a prolog script that retrieved this from .sapconf. We specify a default access of read to the directory and No Access to the files within it; the same definition is created for the /usr/sap/${SID}/SYS directory and files:

```
/usr/sap/${SID}/DVEBMGS${INSTANCE}  = [R] audit(RN)  ;
/usr/sap/${SID}/DVEBMGS${INSTANCE}/*  = [N] audit(RN)  ;
/usr/sap/${SID}/SYS  = [RX] audit(RN)  ;
/usr/sap/${SID}/SYS/*  = [RX] audit(RN)  ;
```

We lock out the /usr/sap/trans transaction directories with default read access to the directories and No Access to the files. However, the .sapconf file needs to be readable by multiple users; so, we add a specific resource record for that.

```
/usr/sap/${SID} = [RX] audit(RN)  ;
/usr/sap/trans  = [RX] audit(RN)  ;
/usr/sap/trans/*  = [N] audit(RN)  ;
/usr/sap/trans/.sapconf  = [RX] audit(RN)  ;
```

The last resource entry sets the default access to No Access for the home directory of the SAP Administrator ID (note that some installations will not use a home directory):

```
/home/${SID}adm = [N] audit(RN)  ;
/home/${SID}adm/*  = [N] audit(RN)  ;
```

We have not defined any process or TCP services resources in this LDF and did not attempt to define conditional access.

## 4.2 The Tivoli Framework Lockdown Module

In this section, we are going to describe the procedures to implement an LDF on the Tivoli Enterprise Framework working in a UNIX environment. The UNIX Framework LDF implementation is largely a super-set of what you can also achieve for Windows NT. For an example of locking down the Windows NT environment, refer to 4.6, “The Windows NT Operating System Lockdown Module” on page 83.

In investigating the requirements for a Tivoli Framework lockdown module, we found that, unless you have a requirement for extremely restrictive levels of security, there is little that needs to be added through the use of TSM. The important thing is to make use of existing Framework security.

### 4.2.1 Framework Security Considerations

As you may already know, the Tivoli Framework environment has a number of security mechanisms and features, since it has been designed to provide secure and reliable invocation of distributed applications. It provides security capabilities and such role-based administrator services as:

- Installation password that prevents unauthorized client additions to the management network.
- Implementation of authentication and authorization. The system *authenticates* that you are who you say you are, and when an authenticated user attempts to perform some operation, the system verifies that you have the correct level of *authority*. This also applies when any Tivoli application action takes place.
- Strong encryption using DES and the use of Kerberos authentication service.
- Tivoli Administrators. After installing the TMR Server, you can define a Tivoli administrator that can perform functions that require root access but who logs on from a name different from *root.* In this way you won't need to share the root user's password. The Tivoli super user, also known as the Tivoli root user, does not need to have logged in to a machine as root or Administrator. The Tivoli root user can be assigned to any user name using `wauthadmin`.

  Note that, by default, the TMR server installation defines root (or the user that was used to install Tivoli) as the Tivoli root user. Managed nodes do not do this and should not need the root login defined as the Tivoli root user.
- Tivoli administrators can have their capability defined by policy regions and authorization roles which are becoming increasingly fine-grained (such as the Tivoli User Administration Admin_Mod_Password role introduced in 3.6.1). The Tivoli administrator can perform tasks requiring high privilege levels without automatically gaining privilege in undesirable areas.

  Note that there is a distinction between these Tivoli administrator authorization roles and the TSM concept of Role-based security.

In spite of such security features being available, there may be certain especially sensitive environments where you might wish to employ additional security protection and auditing.

We assume here, or at least expect, that the network environment where the TMR Server and all Managed Nodes reside is not open to outside attacks or has some measures in place to guard against them. The TMR server must also be regarded as a security-sensitive device and placed where only authorized people have physical access to it. It is very important that the password of the root (UNIX) or Administrator (Windows NT) user ID in the TMR Server should be known only by appropriate Tivoli support and administrative personnel. The recommendation is that the TMR server is dedicated to the Tivoli environment thus reducing any requirement for those not involved in systems management to have access to the machine.

In a general view, our security scheme is aimed at avoiding unauthorized access to the resources available in a typical Tivoli environment installation. Such resources include binaries, configuration files, log files, audit files, processes and daemons, communication between the TMR Server and its respective Managed Nodes and Endpoints, protecting them against data corruption, modification, halting and TCP port interception.

### 4.2.2 TSM Roles and Groups in the Framework LDF

We created four new roles, and, to show examples of their use in groups, we also created sample groups. These appear in the LDF as follows:

```
GROUPS
   Admin_kill "Admin_kill" = (shamu)  audit(LF,RF) ;
   Administrators "Administrators" = (root)  audit(LF,RF) ;
   Backup_operators "Backup_operators" = (back_op1)  audit(LF,RF) ;
   Super_TACF "Super user for TACF" = (tmesec)  audit(LA,RA) ;
ROLES
   Role_of_Super "Role_of_Super" = (Super_TACF) ;
   Role_of_Administrators "Role_of_Administrators" = (Administrators) ;
   Role_of_Backup_operators "Role_of_Backup_operators"=(Backup_operators);
   Role_of_admin_kill "Role_of_admin_kill" = (Admin_kill)
      parent(Role_of_Administrators) ;
```

We declare a few sample users in these groups to demonstrate likely allocations of users to groups and roles. The shamu and back_op1 user names were chosen as examples and, as with all the lockdown modules, the group definitions will need modification for your own environment.

The roles were determined as follows:

**Role_of_Super**
This role contains all resources defined in the LDF and is assigned *Full Control* access. In our sample groups we gave this role to the *Super_TACF* group and only placed *tmesec* in this group. This is the ID that owns the TACF database and its associated files. This TACF user ID, by default, has the

permissions that allow it full control on all system resources.

Adding a user to the Super_TACF group will give them full control over the resources defined in the LDF. This allows access to the resources without making the person a true TACF administrator. To make them a TACF administrator, they would need to be specifically added as one, for example, by running the Add / Remove TACF Auditors / Administrators job, or by specifying other users as TACF administrators at TACF installation time.

**Note**

It is not good practice to set the root user as a TACF administrator. It would give root the ability to kill TACF processes and change your security policies.

**Role_of_Administrators**

This role is intended for root-level administrators of the machine on which the Tivoli Framework resides, in other words people who need a reasonable amount of control over the machine but do not need access to TACF itself. The Tivoli Framework runs as root itself, and, so, we need to ensure the root ID has the required authority to work with Framework directories.

When the Tivoli Framework executes a method, it will normally be executed with the privileges provided by the operating system. For security purposes, all Tivoli methods normally run as an unprivileged user. On a UNIX system, this is the user *nobody.* A programmer can set a particular privilege for a method, if required, in order to perform system management operations. When the method is invoked, it runs as if a particular privileged user had invoked it.

For the Framework, this privileged user is typically the user *root* or the UID/GID defined for the Tivoli administrator. In our case, we have defined only the root user as a Tivoli Administrator.

This role allows full access to Tivoli resources, however, it still does not allow root user to kill Tivoli processes or to delete database and log files. It also allows Read Only access to TACF resources.

The Role_of_Administrators role has the following access rights defined:

```
ROLE "Role_of_Administrators" =
/etc/Tivoli   [F],
/etc/Tivoli/*    [F],
/usr/lib/X11/app-defaults/Tivoli    [F],
${BINMP}    [F]   ,
${BINMP}/*    [F],
${VARMP}    [F],
${VARMP}/*    [F],
${VARMP}/${HOSTNAME}.db/odb.bdb    [RWU],
${VARMP}/${HOSTNAME}.db/gwdb.bdb    [RWU],
${VARMP}/${HOSTNAME}.db/gwdb.log    [RWU],
${VARMP}/${HOSTNAME}.db/notice.log    [RWU],
${VARMP}/${HOSTNAME}.db/odb.log    [RWU],
${VARMP}/${HOSTNAME}.db/odtrace.log    [RWU],
${VARMP}/${HOSTNAME}.db/oservlog    [RWU],
${VARMP}/${HOSTNAME}.db/oservlog.start    [RWU],
${VARMP}/${HOSTNAME}.db/epmgrlog    [RWU],
${TACFDIR}    [RX],
${TACFDIR}/*    [R],
```

**Note**

We have defined some variables in order to keep the LDF as generic as possible. You may change them according to your installation, prior to the profile manager definition, by editing the FW3.6_UX_LockDown.ldf file. Our variables were defined as follows:

```
VAR
  BINMP=/usr/local/Tivoli;
  VARMP=/var/spool/Tivoli;
  TACFDIR=${BINMP}/TACF;
  INTERP=aix4-r1 ;
  HOSTNAME=dover ;
```

See Appendix B.2.3, “LDF PROLOG Section” on page 113 for a description of the prolog where you can use a script to set variables before the LDF file is interpreted.

**Role_of_admin_kill**

We wanted one group of administrators whose users have the ability to kill sensitive Tivoli Framework daemons.The Group Admin_kill was assigned the role Role_of_admin_kill. Any groups with this role, have the same privileges as those with the Role_of_Administrators role, PLUS the ability to stop Tivoli processes. You should not include the root user in this group. Even if some other user performs an su to a user defined in this group or to root, they will not be capable of killing any Tivoli process, as TACF recognizes the original user ID and refuses the action.

**Role_of_Backup_operators**

The *Backup_operators* Group and the *Role_of_Backup_operators* has been created to grant access to users in charge of backing up the system. Groups that have this role have *Read* access to all resources defined in the LDF. The user *back_op1* is defined as a sample user.

### 4.2.3 TSM Resources in the Framework LDF

This section describes the actual resources protected through the LDF. Note that in the Framework LDF, all resource protections are set to warning mode. This means that TACF will perform the check and report on the defined result but will always grant the access.

#### 4.2.3.1 Files and Directories

Every FILE resource defined in the lockdown file, except the TACF FILE resources, are assigned default access as *Read Only* and audit control on both successes and failures. It is very important to have as much information as possible in case you need to check the TACF log files through the secons utility.The TACF resources are assigned the default access of *None*.

All binaries must be world executable as well as their directories:

```
${BINMP}/bin  = [RX] audit(RA) ;
${BINMP}/bin/*  = [RX] audit(RA) ;
${BINMP}/bin/${INTERP}  = [RX] audit(RA) ;
${BINMP}/bin/${INTERP}/*  = [RX] audit(RA) ;
${BINMP}/lib  = [RX] audit(RA) ;
${BINMP}/lib/*  = [RX] audit(RA) ;
```

**Note**

There are some resources we do not want to grant read and execute only access to. For example, the tasks and UserLink data stored in $BINDIR/TAS need to be writeable.

During the installation time and during task creation or modification, the write and update permissions are used. In our test environment, that responsibility was assigned only to the root user. You might define other users to perform this kind of operation, however, this user should be in a group with the Role_of_Administrator role in the LDF.

#### 4.2.3.2 Processes and Daemons

We have also defined both TACF and Tivoli Framework main processes and daemons with default access of No Access protecting them from being killed. We have considered the case where we have the TMR Server acting as both Endpoint Manager and Endpoint Gateway.

```
PROCESS:${TACFDIR}/bin/seagent "TACF process" = [N] audit(RA) ;
PROCESS:${TACFDIR}/bin/seosd "TACF process" = [N] audit(RA) ;
PROCESS:${TACFDIR}/bin/seoswd "TACF process" = [N] audit(RA) ;
PROCESS:${BINMP}/bin/${INTERP}/TMF/LCF/ep_mgr = [N] audit(RA) ;
PROCESS:${BINMP}/bin/${INTERP}/TMF/LCF/gateway = [N] audit(RA) ;
PROCESS:${BINMP}/bin/${INTERP}/bin/oserv = [N] audit(RA) ;
```

Note that the built-in Framework authentication prevents unauthorized users from stopping these daemons through Tivoli commands such as `odadmin shutdown` or `etc/Tivoli/oserv.rc stop`.

#### 4.2.3.3 Network Communication Resources

In a typical TME installation, three types of network servers are found:

- The Object Dispatcher (oserv) on all Managed Nodes and Endpoint Gateways
- The Tivoli Management Agent gateway daemon (gateway) on all Endpoint Gateways
- The TMA daemon (lcfd) on all TMA Endpoints

The built-in security of the Framework includes a number of features to prevent security problems with these services. In order for actions to be performed, the principal initiating the action has to be authenticated and authorized. There are also a number of actions that can be taken to reduce the openings for an attack. These include oserv settings through the `odadmin` command:

**allow_client_install**
This can inhibit the addition of managed nodes.

**set_allow_rconnect**
This determines whether or not administrators can connect to this oserv using the Tivoli Desktop for Windows.

**set_install_pw**
This sets a password preventing a Tivoli administrator from installing new products unless they know this password.

**set_port_range**
This sets the range of ports the oserv will use in addition to port 94.

TMA endpoint connection rules can be determined through gateway policies.

If we wanted to implement further protection using TSM, we need to remember the following: all managed nodes (including gateways) may potentially communicate with each other and must communicate with the TMR Server. Managed nodes may also need to communicate with nodes in another TMR. TMA endpoints must be able to communicate with their gateway, but it is not necessary for them to communicate with the TMR server or other endpoints (from a Tivoli perspective).

The oserv daemon coordinates communications between the TMR Server and all managed nodes and gateways. In a normal situation, It keeps the TCP port 94 continuously open. However, if the data to be transferred is larger than 16K bytes (alterable since 3.2), oserv establishes Inter Object Messaging (IOM) and bulk data transfer (BDT) and allocates the next port number available based on ephemeral ports and any port range set on the oserv with `odadmin set_port_range`. The communication between TMA endpoints and their gateway is done by using the TCP port 9494 on the gateway and the TCP port 9495 on the TMA endpoint (Tivoli Framework version 3.6 and 3.2 used TCP port 9494 on the Endpoint by default). Endpoint to gateway communications does not use BDT/IOM.

We also looked at Inter-TMR communications. Multiple TMR servers can be linked together and the communication between them is also done over TCP port 94.

In our LDF, we locked the TCP port 94, TCP port 9494 and the TCP port 9495. Since the TCP port 94 is referred to in the /etc/services file as objcall, we had to define the entry in the LDF by its name.

```
TCP:9494 "Lock TCP port 9494 " = [N] audit(RA)  ;
TCP:9495 "Lock TCP port 9495 " = [N] audit(RA)  ;
TCP:objcall "Lock TCP Port 94 " = [N] audit(RA)  ;
```

**Note**

This port protection is included as an example of what can be done. Due to the built-in protection provided by the Framework, we recommend the removal of this type of port protection for most installations. You should include this if you are concerned about certain types of attacks, such as denial of service port attacks.

After running the Framework Lockdown script and before distributing the profile, you would need to open the Framework Lockdown Security Profile and check the IP addresses defined to have access. Alternatively, you could add a prolog script that determines these addresses and provides them to the rest of the LDF as a variable. See Appendix B.2.3, “LDF PROLOG Section” on page 113 for more details. They MUST include the IP addresses that need to connect to the following ports:

To port 94:

- The TMR Server, all managed nodes and gateways you have defined in your Tivoli environment will need access including all managed nodes from interconnected TMRs.
- All machines that will need access to the TMR using the Tivoli Desktop.
- All machines that should have access to the TMR using WEB Interface.

To port 9494:

- All TMA endpoints that could potentially use the gateway must have access to 9494 on the gateway.
- All gateways will need 9494 access to 3.6 or older TMA endpoints.

To port 9495:

- All gateways need 9495 access to any TMA endpoint they could potentially be called upon to service.

**Note**

With this configuration, for every new managed node, gateway, or TMA endpoint you add to your Tivoli environment, you must first include its IP address in the TCP ports Access list as appropriate and redistribute the security profile.

## 4.3 The Lotus Domino Lockdown Module

Lotus Domino is a good example of an application for which we can provide a meaningful lockdown module. You generally want to be sure that the data files and logs are not accessed by anyone other than authorized administrators. You also want to be sure that the Lotus daemons are not accidentally killed. The information in this section details our considerations when developing the sample Lotus Domino LDF.

Note that Lotus Domino has its own security built-in such as user authentication. The aim of the module is to protect Domino resources even further. Direct management of Domino, including the monitoring of processes and the management of events, can be achieved through Tivoli using the Tivoli Module for Domino.

### 4.3.1 Lotus Domino Considerations

Our design of the Lotus Domino Lockdown Definition File was based around Lotus Domino version 4.6 running on AIX version 4.2.1. The resources to be protected, the directory structure and running daemon processes, are the same for Lotus Domino 5.0.

Development of this LDF depended on a Domino administrator providing some basic information. This is how we expect modules can be developed in production environments. The Tivoli Security Operator doesn't necessarily need to know how to install and operate the application. What he needs to know in order to create a sufficient lockdown module is just a basic idea of how the application works, how to start and stop it, and how to test normal operation while tracing with the TACF log.

### 4.3.2 Design Considerations of the Lotus Domino LDF

Discussions with a notes administrator revealed that there are two basic UX:FILE resources to be protected.

The first of these is the /dominodata directory. Access to this directory should only be granted to one of the several server processes. One exception to this is the `notes.ini` file. This has to changeable by the notes administrator in order to properly administer the domino server. It will also be backed up by the server process. Therefore, we have specified a resource definition for /dominodata/notes.in*.

The second important resource is /opt/lotus or, as in our case, linked to /usr/domino/lotus, the /usr/domino directory. The domino binaries to be found in this directory only need to be executed by the domino administrator while starting the server. If, on other platforms, the need for version updates arises, you could implement another role with full access to this resource like Role_of_DominoUpdate with resource rights of /usr/domino/* [F].

There are a number of Domino daemon processes that we specify in Tivoli resource records of type **UX:PROCESS**. These must have full access to the /dominodata directory, and they themselves should also be protected by TSM. Note that in customized installations, there might exist other domino processes besides the basic server processes. These would need to be added to the lockdown file.

### 4.3.3 TSM Roles and Groups in the Lotus Domino LDF

For normal domino server operations, we found it necessary to grant the server control capabilities to the domino administrator. This includes server start-up and shutdown as well as some more fine-grained control through the `notes.ini` file. Also, the administrator has to be able to control certain Domino processes such as the ability to kill these processes if necessary. However, the domino server administrator should not have the right to access any files within the /dominodata directory for security reasons; so, we protect the databases against unwanted modifications.

Our LDF defines a number of security roles that can be allocated in your group structure to those security groups that match the corresponding job tasks. For example, if you have a group responsible for system maintenance, you will probably want to add the *Role_of_SystemBackupDomino* to that group. These roles are described below. However, we have also defined a security group. This group has one member that we defined in Tivoli User Administration prior to the Domino installation. The user in our group will be the TMEUserMember *notes* with UNIX group *notes*. These user and group records represent the user-ID and group-ID that the server processes are running under. The security group definition in the LDF is as follows:

```
GROUPS
   Domino_Administrator “Domino Administrator” = audit(LN,RN) ;
```

This user itself was defined in the Tivoli User Profile named Domino4.6_LockDown_UP which resides as well as the Tivoli Security Profile Domino4.6_LockDown_SP within the Tivoli Profile Manager Domino4.6_LockDown_PM.

The following Security Roles were defined in the LDF:

```
ROLES
   Role_of_DominoServerControl “Role of Domino Server Control” = (Domino
   Administrator) ;
   Role_of_DominoWebAdmin “Role of Domino Web Admin” = (Domino
   Administrator) ;
   Role_of_SystemBackupDomino “Role of System Backup for Domino Resources”
   = (“AIX4.2_LockDown_SP:System_Administrators”) ;
```

The *Role_of_DominoServerControl* reflects the above-mentioned resource Domino administrator.

Furthermore, we defined the role of Domino Web administration named Role_of_DominoWebAdmin. This role is necessary to administer the Domino server through the web-interface and it grants the following resource rights:

```
ACCESS
ROLE “Role_of_DominoWebAdmin” = /dominodata/domino [RX],
   /dominodata/domino/* [F];
```

The first line of resource rights is necessary to provide the ability to change the directory to /dominodata/domino, and the second line gives full access to the server web-interface binaries contained in this directory.

The last role, named *Role_of_SystemBackupDomino,* grants some resource rights to the AIX backup facility. These resources are only accessible with conditional rights:

```
ACCESS
ROLE “Role_of_SystemBackupDomino” =
   /dominodata/* ,
   /dominodata/* /usr/sbin/backbyname [R] ,
   /dominodata/* /usr/bin/find [RX] ,
   /dominodata/domino [RX] ,
   /dominodata/domino/* ,
   /dominodata/domino/* /usr/sbin/backbyname [R] ,
   /dominodata/domino/* /usr/bin/find [RX] ,
   /usr/domino/* ,
   /usr/domino/* /usr/sbin/backbyname [R] ,
   /usr/domino/* /usr/bin/find [RX] ;
```

This role is intended to enable the AIX system backup facility to create a bootable system backup image. We don’t intend to use this role to backup domino server data and, therefore, we don’t need a restore role.

Note that AIX system backups result in bootable system images which will restore the complete system as it was at the time of the backup.

### 4.3.4 TSM Resources in the Lotus Domino LDF

The resources we need to take care of in this lockdown module are mainly Lotus Domino server processes. This is why we have 19 processes defined and only some basic directories in our implementation. These processes are controlled through the main server process, which itself is started by the domino start-up script. The data resources we protect are as follows:

```
RESOURCES
/dominodata/* "Domino Data" = [N]  ;
/dominodata/domino "Domino HTML Directory" = [RX]  ;
/dominodata/domino/* "Domino HTML Content" = [R]  ;
/dominodata/notes.in* "Notes Initialization File" = [R]  ;
```

Then, we ensure only read access to the domino binaries and processes:

```
/usr/domino/* "Domino Binaries" = [R]  ;
/usr/domino/lotus/bin/server "Domino Startup Script FILE" = [R]  ;
/usr/domino/lotus/notes/latest/ibmpow/adminp "Domino Process" = [R]  ;
/usr/domino/lotus/notes/latest/ibmpow/amgr "Domino Process" = [R]  ;
```

```
... and so on ...
```

Then, we specify the default access for the PROCESS records such as:

```
PROCESS:/usr/domino/lotus/bin/server "Domino Startup Script PROCESS"
= [N] ;
PROCESS:/usr/domino/lotus/notes/latest/ibmpow/adminp "Domino Process"
= [N] ;
PROCESS:/usr/domino/lotus/notes/latest/ibmpow/amgr "Domino Process"
= [N] ;
... and so on ...
```

## 4.4 The Netscape Enterprise Server Lockdown Module

This lockdown was designed around Netscape Enterprise Version 3.6. Most of the content will apply to earlier releases and is likely to continue to apply to newer releases. The interesting thing about this LDF is that it could also apply to many other products from the same company.

### 4.4.1 Netscape Enterprise Server Considerations

Because of the common Netscape Server architecture, many parts of this lockdown module will apply to other Netscape Server products such as the Proxy Server or the Fast-track Server.

This is due to the separation of the Administration Server and the application server(s) which run as different processes and even under different user IDs. To be more specific, Netscape always uses the same Administration Server in its products; this is great in terms of security administration since we essentially have reusable lockdown modules.

The Administration Server normally has to run under the root user ID in order to properly control the application server(s). This means you can have more than one application server at a time that will be generated, started, and stopped by the Administration Server and that normally runs under a user ID with only restricted operating system rights like *nobody* or, as in our case, under the limited rights of the user *netscape* and group *netscape*.

The operating system user ID and group ID has to be created before the actual product installation. In our case, this was done with Tivoli User Administration and, therefore, resulted in a User Profile named NS_EPS3.6_LockDown_UP that also resided as the Security Profile named NS_EPS3.6_Lockdown_SP in the NS_EPS3.6_LockDown_PM Profile Manager.

### 4.4.2 Design Considerations for the Netscape LDF

While looking at the Netscape Enterprise Server resources we found that the application home directory is a good starting point on the search for resources to be locked down. So, you'll see in 4.4.3, "TSM Resources in the Netscape LDF" on page 75 our first resource to be in the ldf-file was the application home directory /usr/netscape/suitespot itself. We applied the default access of read [R] to this directory which will be enhanced using roles to allow changing into this directory and other actions as appropriate. To give basic protection to other application resources, we applied read access to all resources contained in the /usr/netscape/suitespot/* directory.

We considered granting no [N] rights at all at this stage, but then every single resource the application would need to read would have to be listed in the resource list separately for role-based rights. Also, the UNIX permissions that are not changed by TSM ensure a certain level of security. We also need to consider the need for read access for backup purposes.

This was the basic step and it covered all UX:FILE based resources. To be more specific for other resources and to have a proper resource list to apply role-based resource rights, we further investigated the resources contained in the application home directory. This is described in more detail in the next section.

### 4.4.3 TSM Resources in the Netscape LDF

We started with the Administration Server directories, which, for obvious reasons, have to be in the resource list for later role-based protection. At this point, we discovered that the directory structure of the Netscape Enterprise Server isn't well separated from a security perspective. Log files are mixed with configuration files, and binaries for the Administration Server are in the same subdirectories as the application servers.

```
RESOURCES
/usr/netscape/suitespot "NS EPS Home Directory" = [R]  ;
/usr/netscape/suitespot/* "NS EPS Home Directory Content" = [R]  ;
/usr/netscape/suitespot/*-admin "NS Admin Server Control" = [R]  ;
/usr/netscape/suitespot/admin*/* "NS Admin Server Conf/Log/ACL" = [R]  ;
/usr/netscape/suitespot/admin-serv/logs "NS Admin Server Log Directory" =
[RX]  ;
/usr/netscape/suitespot/admin-serv/logs/* "NS Amin Server Log Files" = [N]
;
```

Note that, in some cases, we use wildcards in directory names. This is to avoid a dependence on specific installations. Netscape uses the host name in these places.

Besides the Administration Server, we discovered some resources for the application server, the daemon binaries, log file directories, and some database directories.

```
/usr/netscape/suitespot/alias/* "key-aliases" = [R]  ;
/usr/netscape/suitespot/bin/admin/??/*.db "*.db" = [R]  ;
/usr/netscape/suitespot/bin/admin/ns-admin "NS Admin Server FILE" = [R] ;
/usr/netscape/suitespot/bin/admin/ns-cron "NS Cron Daemon FILE" = [R]  ;
/usr/netscape/suitespot/bin/https/admin/bin/* "db, mime" = [R]  ;
/usr/netscape/suitespot/bin/https/ns-httpd "NS WebServer FILE" = [R]  ;
/usr/netscape/suitespot/bin/https/uxwdog "NS Watchdog Daemon FILE" = [R]  ;
/usr/netscape/suitespot/docs/* "NS docs Directory" = [R]  ;
/usr/netscape/suitespot/https-*/* "WebServer Directory" = [R]  ;
/usr/netscape/suitespot/https-*/logs/* "NS WebServer Log Files" = [N]  ;
/usr/netscape/suitespot/manual/* "Manuals" = [R]  ;
/usr/netscape/suitespot/plugins/* "NS EPS Plugins" = [R]  ;
/usr/netscape/suitespot/userdb/* "NS User DB" = [R]  ;
```

Note that, besides the daemon binaries, we also included the daemon processes as UX:PROCESS resources in our list as well as the Administration Server UX:TCP port giving only predetermined workstations the right to access the Administration Server for control and modification purposes.

```
PROCESS:/usr/netscape/suitespot/bin/admin/ns-admin "NS Admin Server
PROCESS" = [N]  ;
PROCESS:/usr/netscape/suitespot/bin/admin/ns-cron "NS Cron Daemon
PROCESS" = [N]  ;
PROCESS:/usr/netscape/suitespot/bin/https/ns-httpd "NS WebServer
PROCESS" = [N]  ;
PROCESS:/usr/netscape/suitespot/bin/https/uxwdog "NS Watchdog Daemon
PROCESS" = [N]  ;
TCP:8081 "Admin Server Port" = [N]  pass(TCPAccess:"$TCP8081HOSTS") ;
```

Note that we need to pass this script a list of hosts that should have access in the $TCP8081HOSTS variable.

### 4.4.4 Further Design Considerations of the Netscape LDF

Apart from the basic resource considerations, there are some more resources and relations to be taken care of.

In terms of active daemons, we already discovered the Administration Server process as well as the Application Server process(es). There are two more processes which become active when special tasks within the Netscape implementation have to be scheduled. These are the ns-cron (Netscape cron) daemon and the uxwdog (watchdog) daemon.

Unfortunately, most resource accesses are not handled by the daemon processes themselves but through many tool programs which will be launched by the main daemons. This makes it very difficult to apply proper conditional rights within the role-based resource rights. We tried experimenting with conditional access rights for the Administration Server tasks but decided against pursuing this for the application server after finding it nearly impossible to discover all the tools the application server is using for its configuration features. Examples of the conditional access rights we gave through roles can be found in 4.4.5, "TSM Groups and Roles in the Netscape LDF" on page 77.

More necessary resource accesses were discovered after applying the default resource rights and the basic role-based access rights. The only way to find these resource accesses is to trace using the TACF log file either for warning mode messages or for deny statements if not in warning mode. Note that many of the Netscape administration/configuration tools can become unstable if TACF denies them resource access. They then have to be killed manually. Therefore, we recommend using the TACF warning mode while exploring the range of the applied access rights for any application or while testing this LDF in your environment.

### 4.4.5 TSM Groups and Roles in the Netscape LDF

This lockdown definition file comes with a simple group example structure containing three groups:

```
GROUPS
Netscape_EPS_Administrators "Netscape EPS Administrators" = audit(LF,RF) ;
Netscape_EPS_Auditors "Netscape EPS Auditors" =   audit(LF,RF) ;
Netscape_EPS_WebAdministrators "Netscape EPS WebAdministrators" =
audit(LF,RF) ;
```

Your own group structure might attach roles to more generic groups. For example, a group named HQ_Server_Administrators might manage all server applications; so, they would have something like the Role_of_AdminServerControl (see below).

We introduce four Roles, two of them granting accesses for the `Netscape_EPS_Administrators` Group:

```
ROLES
Role_of_AdminServerControl "Role of AdminServer Control" =
(Netscape_EPS_Administrators) ;
Role_of_WebContentAdministration "Role of WebContent Administration" =
(Netscape_EPS_WebAdministrators) ;
```

```
Role_of_WebServerAdministration "Role of WebServer Administration" =
(Netscape_EPS_Administrators) ;
Role_of_WebServerAudit "Role of WebServer Audit" = (Netscape_EPS_Auditors)
;
```

**Role_of_AdminServerControl**

This role is intended to grant the necessary rights to start, stop, and restart the Administration Server using three scripts named start-admin, stop-admin, and restart-admin. In the ACCESS section, we define this as follows:

```
ROLE "Role_of_AdminServerControl" = /usr/netscape/suitespot/*-admin [RX];
```

**Role_of_WebContentAdministration**

The main purpose of the Netscape Enterprise Server is to serve clients with the requested web-content. This content needs to be administered as well. In larger implementations, it might be useful to introduce some more specific roles for this content administration area while applying rights to a more structured web-content directory:

```
ROLE "Role_of_WebContentAdministration" = /usr/netscape/suitespot/docs/*
[F];
```

**Role_of_WebServerAdministration**

Besides the right to start, stop, or even kill the daemons, this role grants four main access blocks to the administrator. The first are the administration rights for the Administration Server itself - implemented with conditional access rights and preventing any unauthorized configuration for this part of the application being run under the root user ID. Second, we added the necessary conditional rights to the /usr/netscape/suitespot/alias/* resources containing the authentication keys and key aliases. Third are the conditional rights to the application server log files to prevent tampering with these security-relevant files that might contain user-related information as well. And, finally, we add the necessary conditional rights to control and maintain the Netscape user database if kept in a local LDAP structure. (See the LDF for the full role definition.)

Note that, in this more general implementation, some specific resources require conditional access rights (such as /usr/netscape/suitespot/admin*/* /usr/bin/mv [F] ). These native unix programs are mostly used to control log files, to create them if necessary or to rotate them if specified. To prevent the administrator from using these files for administrative tasks not intended by the developers, one might want to investigate the application in more detail. This results in a longer resource list, more role-based conditional rights, and the potential for problems when upgrading to newer versions. Remember, the normal UNIX rights exist to provide some protection for the operating system resources.

**Role_of_WebServerAudit**
A group with this role is able to administer the Administration Server log file thus providing the ability to separate general server administration from auditing. This is achieved with a simple definition as follows:

```
ROLE "Role_of_WebServerAudit" =
/usr/netscape/suitespot/admin-serv/logs [RX],
/usr/netscape/suitespot/admin-serv/logs/* [F];
```

## 4.5 The Oracle Lockdown Module

Relational databases are a common choice for security protection. You generally want to be sure that the database files and logs are not tampered with by anyone other than authorized administrators. You also want to be sure that the database daemons are not accidentally killed. The information in this section details our considerations when developing the sample Oracle LDF. This can be a useful starting point for considering other databases and similar applications.

Note that most RDMBSs have some form of security built-in such as user management. The aim of the module is to protect database resources even further.

### 4.5.1 Oracle Considerations

The first goal of the Oracle Lockdown is to allow only database administrators (DBA) full access to the resources that compose the Oracle server. Oracle operates in a fairly simple manner, for example, in its use of processes.

Potential resources for protection can be divided into the following groups:

- Database files
- Audit logs
- Configuration files
- Binaries

Most of the resources are also grouped into different directories according to their general purpose.

The directory structure for Oracle database installations is set by the Oracle Financial Analyzer (OFA) standard. The Lockdown module considers the OFA standard and has been designed to adapt to different options. This directory structure ensures the grouping of the resources and makes it easy to lock down entire directories within the installation. OFA assumes that all directories are created under a group of mount points.

All the Oracle programs run under the *oracle* account. This user is also the owner of all the files under the OFA mount points. So, this user installs the software and runs it. Some activities require the root account during installations. The Lockdown Module is designed to be applied after the database software has been installed.

Oracle itself runs as a set of daemons. On the process list, each instance of the daemon has a different name, but, the binary is the same for all of them (the oracle executable in the bin directory). All access to the database files is done by this program; so, other users and programs do not need direct access to these files. Starting and stopping the database involves another program called svrmgrl; this program starts the `oracle` program.

Most programs have to read configuration files that exist within the OFA directory structure. These files should only be updated by the DBA since erroneous changes may cause problems in the operation.

The heart of Oracle lies in the data files and the control files. Control files are used to define how to load the databases, and the data files are the data that composes the database. Once these have been loaded, everything is handled by Oracle-specific authentication and security schemes.

### 4.5.2 TSM Roles in the Oracle LDF

In the operation of an Oracle database, there are several job tasks and, therefore, roles that can be easily identified. They correspond to the main activities performed when operating the database:

| | |
|---|---|
| **Administration** | This involves changing configuration files and modifying the way the database works. This also involves installing and updating software. |
| **Backup** | Backup is done in two ways in Oracle. Incremental backup is done by copying the archive (transaction) logs. The full backup is done by copying the data files on an instance that is not operating. |
| **Restore** | Restoring involves copying back data files and then applying transaction logs (archive logs) on the restored files. This is normally done from outside the daemon processes using OS-specific copy mechanisms. |

| | |
|---|---|
| **Operate** | Starting and stopping the database is required here. It may be necessary to stop the database in order to change the configuration or to perform a backup, stop the host, and so on. |
| **Use** | This is the catch-all class of activity; most really important activity is done here but only using the database programs to access data files. Essentially, all that is needed is to connect from the database daemon, and it will handle the actual operation. |

This may not be the extensive list of activities and roles needed, but it covers most of the activities to be dealt with from the TSM point of view.

### 4.5.3 Goals of the Oracle LDF

The Oracle Lockdown was designed to allow the Oracle database to continue its normal operation while protecting the data files from other users including root. The *oracle* user account is assumed to be the one on which the daemon is executed.

Backup and restore operations must be possible, otherwise, the database directory should be restricted to access only through the daemon. Configuration files should be protected so that only a valid database administrator (DBA) can handle them. An operator must be capable of starting and stopping the database. Installing new binaries must be possible, but only a DBA is allowed to use them.

The *oracle* user should not be used as a work account. DBA and operators should log in using their normal account and then switch to the oracle user. This allows the server to run on the oracle account while maintenance tasks are handled by other accounts. This allows a finer control to be granted to the oracle account. Under TACF, authorizations for UNIX user activities will be checked against the account they logged on this.

The purpose of this design is to ensure that only database administrators have complete access to the data files. These files should only be handled by the daemon, and, thus, having other users (such as root) accessing them can be potentially dangerous. Oracle’s default installation is already reasonably secure. The goal of this added level of security is to avoid accidental shutdown of the instance or damage to the data files.

### 4.5.4 Design Considerations of the Oracle LDF

Determining what to lock down was done by analyzing the TACF trace logs. This allowed us to determine which resources where used and what kind of access was needed. It also allowed an easy verification of the resource names. Since the resources include many wild cards, being sure the correct resources are being protected is crucial. To achieve this, TACF was placed in warning mode to test the module.

The trace file included all information necessary to determine the resources. It includes what program, what file, what kind of access, what rule (resource name) was used, and if it was accepted or not.

To determine exactly what was being treated, we stripped all the irrelevant information from the trace output. The result was then sorted out and analyzed. The analysis of the file accesses alone allowed us to determine how the Oracle Database starts up.

The next step involved determining which roles would be created. This process is somewhat difficult because there are no set rules for the definition of roles. A simple approach is to make two levels: users and administrators. This is a valid set of roles but lacks the refinement that may be necessary in a complex environment.

The final choice was for a more divided structure. In a database, we have the programs that service the clients, the administrators, and the users as basic roles. Also commonly used are roles for backup and restore of the database as well as starting and stopping the service. These were listed above and chosen as the ones to be created.

Users will have the default access. There was no intent of eliminating any user from using the client part of the server. The reason is that Oracle itself can filter out undesired users (by means of its internal user authentication). So, default access to the resources should allow anyone to use the client part.

The other tasks require more access to the datafile, audit, and configuration directories. Each of them has a different access level. Backup can read the datafiles, and restoring it may be necessary to write to the directory. The write access is necessary because backup in Oracle can be done by copying the datafiles. If this kind of external backup is not used, the role of restore should not be used.

Administration was granted full control to the binary, audit, and configuration files. Configuration and binaries must be accessed by the DBA, but the audit files can be protected. Whether to protect them or not is a site specific issue and the general LDF does not protect them from the DBA. One reason is that the DBA is normally the person best suited to study these files, but this may not be the case in all places.

One interesting effect of protecting the audit logs is that, if the user cannot write to the logs, he cannot start or stop the database service. Service start-up and shutdown is handled by a server manager application (named svrmgrl). This program has to be able to write an audit trail; if it cannot, it will not work. As a result of this characteristic and the design of the LDF files, only Operators can start and stop the database (DBA is a child role of Operators; so, a DBA can also start and stop the database).

The daemon that runs the service can be started by any user in the operator group. But, due to Oracle's design, the service runs under the name of the user that started it. So, to avoid the server running in a user account the operator must switch to the *oracle* user prior to starting the server.

This also means that anyone that starts the server must have the same conditional access to the data files as the oracle user. This is a problem in the lockdown design. It is impossible to completely isolate the oracle user since it is needed to effectively run the database. The oracle user is granted a DBA access to the resources. This is not a severe restriction because, normally, the DBA will log in as the oracle user to perform activities. But, attempts to design a lockdown module that isolates the oracle user could also be considered.

Start-up of the database was tricky and required many grants to work effectively. Another problem encountered occurs when using the client tools. If the Oracle Listener is not active or the connection is done in a local level (not using the listener), the user must run the daemon process. This is how the client tool works on the same machine. This means that all the users would have to have conditional access to the data files. This is highly undesirable and potentially dangerous. So, the LDF is written to allow connection only through `SQLNet` (Oracle's network connection API). An attempt to do it directly will result in error messages.

## 4.6 The Windows NT Operating System Lockdown Module

In this and other Redbooks concerning Windows NT, we strongly recommend and assume that NTFS is being used versus FAT, a topic discussed in the redbook *Tivoli Security Management Design Guide*, SG24-5101. NTFS brings additional features such as local folder and file-level permissions, ownership, and auditing. An understanding of the Windows NT operating system and its relationship to TSM is the first step in considering resource requirements. This section is not intended to explain in detail the NT operating system or TSM, but, instead, we will look at the capabilities of TSM with NT in order to understand what we need to look at with regards to managing the protection of Windows NT resources. Refer also to 3.3.2, “Windows NT 4.0 Considerations” on page 47 where we discuss other considerations of TSM for NT.

The Tivoli security model with groups and roles maps closely to the one that is implemented in the Windows NT operating system. The Tivoli security groups correspond to the Windows NT global groups, and the Tivoli security roles correspond to the Windows NT local groups. The Administrator account natively has full control of resources and is treated the same with TSM.

In our LDF, the TSM group *Domain Users* and TSM role *Users* are equated to the Interactive NT group Everyone; so, by locking down resources with Default Access, you are locking/changing Everyone’s access to that resource. The overall picture of locking down the Windows NT operating system was from the most restrictive (Users) to the most open (Administrators) with varying degrees in between for the roles, again, because the administrator account must have full control of NT by default, we did not attempt to separate many different administrative abilities. In TSM for Windows NT, it is much easier to add restrictions and child groups and roles than it is to remove them. This lockdown provides protection on a standard domain-structured (versus peer to peer) Windows NT network using native global and local groups and users. In 3.3.2, “Windows NT 4.0 Considerations” on page 47, we give ideas and considerations for the general use of TSM to manage Windows NT access control.

This lockdown file does not include shares or printer resources due to the very site-specific nature of these resources, and note that TSM 3.6 does not manage Native NT shares, that is, Admin shares that contain the dollar sign ($). See also 3.3.2.6, “NT:PRINTER” on page 52 and 3.3.2.3, “NT:SHARE” on page 50.

Every NTFS file and directory has an owner who has a special status. The owner of that resource can manage the permissions of that resource as if they were the Administrator account. They can even deny the Administrator the right to access that resource. The bottom line is that resources created by the user are owned and managed by the user unless the Administrator takes ownership. Windows NT allows you to modify the way Windows NT treats the permissions of file owners (specified through the Windows NT group CREATOR OWNER). TSM 3.6 does not provide any facility to alter the way

Windows NT treats the permissions of the CREATOR OWNER group. If you must modify these permissions, you must use native Windows NT tools.

In creating the lockdown file, we started by populating a profile using the TSM populate utility, populating the groups, roles, and the NT user rights. In a standard installation, populating creates TSM security groups and roles from the NT Global groups and local groups and the TSM security resources as follows:

**Security Groups (NT Global Groups):**
Domain Admins, Domain Users, Domain Guests.

**Security Roles (NT Local Groups):**
Administrators, Backup Operators, Print Operators, Account Operators, Replicator, Power Users, Users, Guests.

**Resources:**
NT:SYSTEM from NT User Rights. In version 3.6.1 NT:PRINTER and NT:SHARE resource types will also populate but we do not include these in our LDF for reasons already stated.

We implement our security management in two stages. First, we secure certain resources (such as sensitive directories, files and some registry entries) with security resource records that define a restricted default access level. Then, we allow specific roles a greater capability over those resources by modifying the NT user rights for those roles. This provides us with the level of protection we desire without having to individually name a large number of file and directory resources to be modified in all the roles.

For the lockdown, we removed the management of the group *Domain Guests* and the role *Guests* from the populated profile; we do not need to manage these in our LDF to achieve what we need. We suggest that depending on the company's security policy and legacy or current architecture, Guest groups and roles should not be managed from TSM. By default, Windows NT has the Guest accounts disabled, and unless the customer/company has allowed guest access, it should remain disabled. If the customer has the guest account enabled, lockdown of the guest role needs to be done according to the requirements that the company security policy has set for that account.

This lockdown also does not manage the roles of *Print Operators*, *Server Operators*, and *Replicator*, since their use is likely to be unique for each production site. We have designed this lockdown file so that if those roles are desirable, modification of this lockdown to accommodate those roles would be easier than to remove those roles and adjust the resources if they are not.

### 4.6.1 TSM Roles and Groups in the Windows NT LDF

The roles we defined and the groups we used to demonstrate them were defined as follows (note we must use double quotes for strings that contain spaces):

```
GROUPS
   "Domain Admins" "Designated administrators of the domain" =
   (newadmin,Administrator)  audit(LF,RF) ;
   "Domain Users" "All domain users" = (newuser,testuser)  audit(LF,RF) ;
ROLES
   "Account Operators" "Members can administer domain user and group
   accounts";
   Administrators "Members can fully administer the computer/domain" =
   ("Domain Admins") pass(NTGroups:"Domain Admins") ;
   "Backup Operators" "Members can bypass file security to back up files" =
   pass(NTGroups:"Domain Users") ;
   "Power Users" "Users with the right to shutdown a system" = ("Domain
   Users") ;
   Users "Ordinary users" =  pass(NTGroups:"Domain Users") ;
```

The roles were determined as follows:

**Administrators**

Groups with this role (such as *Domain Admins* which includes the Administrator user in our example) have access rights determined by the NT:SYSTEM user rights. We grant this role the rights *S RS RA MD MA B RB CT ID TO* which can be interpreted using Table 2 on page 87. This effectively gives users in groups with this role full access to manage the system.

As with other LDF examples, we have not set access time restrictions. Access times can be set at a production site if there are clear working times of the employees, and company policy does not allow work after hours for reasons such as classified working environment, physical security, and so on.

**Users**

Groups such as *Domain Users* that have his role have their access rights determined by the NT:SYSTEM user rights *RA*. This can be interpreted by Table 2 on page 87 and allows the user to access this computer from the network.

**Account Operators**

Groups with this role receive access based on the NT:SYSTEM user rights *RA MD*. From Table 2 on page 87 we can see the *MD* right allows groups with this role to manage the domain by adding workstations and member servers. This is in addition to the same remote access right also given to those with the Users role.

**Backup Operators**

Groups with this role receive access based on the NT:SYSTEM user rights *RA B*. From Table 2 on page 87 we can see the *B* right allows groups with this role to perform backups in addition to the rights of the Users role. Being able to backup the data does not mean they have full access to it. If they were able to restore the data the ACLs for the data will also be restored along with it.

**Power Users**

Groups with this role receive access based on the NT:SYSTEM user rights *S RA*. From Table 2 on page 87, we can see the *S* right allows groups with this role to perform a shutdown in addition to the rights of the Users role.

As you can see, Windows NT operating system resource protection management is achieved with our TSM roles through assigning only one resource, NT:SYSTEM user rights. This is because of the NT built-in (or INTERACTIVE) group Everyone. Individual resources in this LDF need not be assigned to every role. We create the resources in TSM to apply the default accesses which will assign permissions to everyone including the group Domain Users.

We use NT:SYSTEM user rights resource in our roles to modify the rights in a more global fashion. The rights are summarized in the following table (see also the discussion of the system resource type in 4.6.2.1, “System” on page 89):

*Table 2. Windows NT Rights*

| LDF / CLI Option | Name in Tivoli Desktop | Name of Windows NT Right |
|---|---|---|
| S | Shutdown | Shut down the system |
| RA | Remote Access | Access this computer from network |
| MA | Manage Audit | Manage auditing and security log |
| RB | Restore | Restore files and directories |
| ID | Install Devices | Load and unload device drivers |
| TO | Take Ownership | Take ownership of files and other objects |
| RS | Remote Shutdown | Force shutdown from a remote system |
| MD | Manage Domain | Add workstations and member servers to domain |

| LDF / CLI Option | Name in Tivoli Desktop | Name of Windows NT Right |
|---|---|---|
| B | Backup | Back up files and directories |
| CT | Change Time | Change the system time |
| LL | Local Logon | Log on Locally |

**Note**

Users and new users being added to a system should be placed in a group with the User’s role. This ensures they will fall under the default access permissions.

The Windows NT Account Operator role, Backup Operators role, and Power User roles are not assigned to a TSM group because the created resources default access rights will apply to all roles (Windows NT local groups). In this LDF, we do not assign example groups to all the roles available. In a production site, you will have groups based on the company, and certain groups will have these roles.

If you need to protect resources in a more granular fashion than the NT rights system allows, you will need to specifically name those resources in the security role and modify the access permissions accordingly. For example, if you need groups with the Users role to be able to write to a certain directory that normally they would not have access to, you would need to name that directory as a resource in the User’s role and set the permission for the role on that resource to **Write**.

It is not advisable to assign rights and resources to individual users because this defeats the object and benefits of role-based management. It makes administration more difficult and increases the possibility of leaving undesirable access rights with a user. Although it is possible to specify individual users in resource access control in Windows NT, this capability is not exposed through TSM, and we recommend that you avoid doing it.

In a production site, it should be mandatory to place resources in each role to change access permissions and auditing based on the role’s job function and according to what files will need to be secured. In this LDF, no application program files were included, just the operating system files and directories. Resources for each role can be added or altered with protection for applications on a workstation and server.

Our aim in the LDF was to tighten the default accesses on resources and rights for the role *Users*. The *Administrators* role has full control with no restrictions; so, the role of Administrators is effectively exempt from the default accesses we set on the resources. A recommended model for managing Windows NT is that very few people have access to the Administrator (or equivalent) login ID. The tasks people need to perform will depend on the job roles they have that we implement in Tivoli security roles - and, where possible, these roles have their access defined by the NT user rights.

The key to this LDF was modifying the default accesses to certain key resources. This can be used as a base secure lockdown role that can be built upon by granting access at higher levels (such as Power User, Server Operator). In the case of a company that requires the Guest account, even tighter restrictions can be imposed. Default access in NT is the foundation for developing a good security architecture with TSM.

## 4.6.2 TSM Resources in the Windows NT LDF

This section details the actual resources defined and protected in the LDF.

### 4.6.2.1 System

Populating the resources in NT with TSM 3.6 maps the NT User Rights to a Tivoli security SYSTEM resource named *User Rights*. NT User Rights is the core facility for granting permissions to a TSM role. Since this is the basis for how roles are developed in NT as well as in TSM, and since you should only have one resource type of NT:SYSTEM being distributed to any given target, we will explain some more about what the User Rights are, what they can do for TSM, and how important they are in this LDF.

The user rights policies in native NT are accessed using Administrator Tools. The User Rights Policy grants task or job rights as opposed to resource rights (permissions) to the NT local groups (mapped to and managed by TSM roles). Rights such as *Shutdown the system*, *Access this computer from network*, *Add workstations and member servers to domain*, and *Log on locally* are just a few. Part of the TSM product documentation, the *Tivoli Security Management User's Guide*, lists the rights handled by TSM. The base rights that we used for a standard user (through the Users role) was *Remote Access* allowing for login of a user no matter which domain architecture a site was using. Advanced user rights such as *Bypass traverse checking* or *Create a token object*, and *Debug programs* are not managed by TSM. If these rights are required, the NT user rights policy tool must be used.

As stated earlier, the base user role in the LDF is assigned *Remote Access*. Administrators have all rights given to them in Native NT. In the middle levels of access are the roles of Power User which have the file access permissions set by the defaults in the resource records (they are not modified by the role) plus the ability to *Shutdown* the system in the NT:SYSTEM resource record user rights. Backup Operators have the NT:SYSTEM user right of *Backup*. The reason we don't have to give them more than just default access permissions for the files and directories we are protecting in our resource records is because the native NT *Backup files and directories* right overrides access permissions granted to individual files and directories for backup purposes. If this was not the case, we would have to assign them administrator abilities or read access to otherwise protected directories. Account Operators have the added rights to *Manage Domain*.

#### 4.6.2.2 Files and Directories

The files and directories resources are the most time-consuming. This LDF is based on the documentation from Trusted System's NSA study paper, Microsoft Corporation's security paper, and Microsoft's knowledge base for Windows NT operating systems.

We have included part of a Framework 3.6 lockdown in case the Windows NT system is also a managed node. This can be removed if this is not the case. The following directories are assigned to No Access with auditing set to SUCCESS; so, if a user was trying to access the directories/files, it would be logged.

```
DIRECTORY:"c:\Secmgmt"  = [N] audit(RS)
DIRECTORY:"c:\Tivoli"  = [N] audit(RS)
DIRECTORY:"c:\Tivoli\trip"  = [N] audit(RS)
```

With the native NT architecture running services and programs using the internal Windows NT SYSTEM group, denying access to these directories has no effect on Tivoli functions (like oserv.exe and odadmin). We do not modify the Windows NT SYSTEM group in our LDF - TSM does not allow this operation, and any attempt to do this locally should be subjected to rigorous testing.

The nerve center of the Windows NT operating system is the winnt\system32 directory and the files that fall under it.

```
DIRECTORY:"c:\WINNT\system32\"  = [R]
"c:\WINNT\system32\*"  = [RW]
```

The LDF example above shows that the directory is set to Read only, and newly created files will be set to read and write. TSM 3.6.1 has a finer grain of permissions in its ability to manage the directory and file resource types, and it is possible to restrict all existing files in the system32 directory in 3.6.1. The use of the tools in 3.2.1, “Watching Windows NT Applications” on page 32 will be needed if that ability is employed to ensure that you are not denying access to a vital file in the system32 directory.

The RAS system shown here with the NT:DIRECTORY and NT:FILE resources are:

```
DIRECTORY:"c:\WINNT\system32\ras"  = [N]
"c:\WINNT\system32\ras\*"  = [N] audit(RN)
```

They have been assigned No Access due to the security implications of using the Remote Access Service. If you use RAS, this will have to be modified to fit your needs.

Included in the LDF are the registry editing tools, but, as explained in 3.3.2.4, “NT:REGISTRY” on page 50, there are many work-arounds to this, but, for the standard user, it prevents accidental editing by curious accountants and administrative assistants. Refer to 4.6.2.3, “Registry” on page 92 to see how we can protect the registry entries themselves.

```
"c:\WINNT\Reg.reg"  = [N]
"c:\WINNT\Regedit.exe"  = [N] audit(RS)
"c:\WINNT\system32\Regedt32.cnt"  = [R] audit(RS)
"c:\WINNT\system32\Regedt32.exe"  = [N] audit(RS)
"c:\WINNT\system32\Regedt32.hlp"  = [R] audit(RS)
"c:\WINNT\system32\Regsvr32.exe"  = [N] audit(RS)
```

We have protected the .ini files from users trying to change setup configurations to fit a possible back-door at a later time; this also prevents accidental damage of the files from inexperienced operators.

```
"c:\WINNT\system.ini"  = [R] audit(RS)
“c:\WINNT\win.ini - [R] audit(RS)
```

The standard network executables for accessing other systems and for investigating or scanning the network are the next items protected.

```
"c:\WINNT\system32\FTP.exe"  = [N] audit(RS)
"c:\WINNT\system32\Finger.exe"  = [N]audit(RS)
“c:\WINNT\system32\telnet.exe” = [N] audit(RS)
```

In most situations, none of these tools are required to do a standard user’s job. Users who are serious about executing most of these types of files can go to the Internet and download similar programs which will give them the same functions. TSM 3.6.1 has a finer grain of ability to prevent execution of newly created files and directories that could also prevent this addition of outside tools from happening.

Familiar to most users, the following files are usually accessed from the main window using the Start button or through the CTL-ALT-DEL key sequence. Setting the default accesses on these files prevents the standard user from accessing the Task Manager from the Command line.

```
"c:\WINNT\system32\cmd.exe"  = [N] audit(RS)
"c:\WINNT\system32\taskmgr.exe = [R] audit(RS)
"c:\WINNT\system32\taskman.exe = [R] audit(RS)
```

The next resources are UNIX-style commands.

```
"c:\WINNT\system32\rcp.exe"  = [N] audit(RS)
"c:\WINNT\system32\rexec.exe"  = [N] audit(RS)
"c:\WINNT\system32\rpcss.exe"  = [N] audit(RS)
"c:\WINNT\system32\rsh.exe"  = [N] audit(RS)
```

Restricting the use of these files prevents work-arounds to accessing other machines especially UNIX boxes. In UNIX, the Tivoli security resource UX:TCP is used to restrict access to other machines using TCP, and UX:CONNECT restricts access through programs like these. We do not have the same resource type currently for Windows NT; so, to prevent access, we have locked these down as individual files.

#### 4.6.2.3 Registry

To protect the registry from users, we have attempted to remove access to the various registry editing programs using files resources in the previous section. However, as mentioned earlier, for someone who wants to modify the registry, locking them out of editors will not be sufficient. This LDF assigned Read only access on some of the more important hives and keys in Windows NT. The LDF did not attempt a lockdown of any program's or application's registry keys.

Windows NT does protect itself from even the administrator accounts on certain important keys such as HKEY_LOCAL_MACHINE\Security which stores the security information and HKEY_LOCAL_MACHINE\SAM which contains the user and group security information. Important keys to protect are HKEY_LOCAL_MACHINE\Software (by default Everyone has Read access only) and HKEY_LOCAL_MACHINE\System (again, Read by Everyone is the default).The LDF is just a sample collection of what should be locked down as a minimum.

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion contains several hives that we found to be of value in applying a read-only access permission to a standard user. At a production site, It is recommended that you create resources with default values even if NT already contains that same permission by default. This is the case in many of the registry entries we defined in the LDF. The reasoning behind this is that in a large environment, changes from administrators, whether intentional or unintentional, can change defaults to open up resources without being noticed. If a profile contains the resource, upon distribution, any NT default resource that may have been changed can be set correctly through a profile distribution.

## 4.7 The AIX Lockdown Module

Securing operating systems requires much more work than just locking down a single application. There are several aspects you must keep in mind, especially in the case of UNIX operating systems where we implement a second layer of security - the Tivoli Access Control Facility (TACF).

We can use the fact that the native operating system security is still operative, and we can use TACF to implement stronger, consistent, role-based resource access rights. The important thing to avoid is the disruption of normal operating system operations. Therefore, the development of an operating system lockdown module is a longer process that should include continued testing and monitoring after implementation. The TACF log file should be used extensively to watch for operating system internal problems that might not be noticed otherwise. We recommend the use of the forwarding of serious TACF errors to the Tivoli Enterprise Console once the implementation is in use.

While UNIX operating systems can be regarded as generally secure, they have one real problem in the case of distributed administration - the special position of the *super user* root. Once an administrator has to log on as root in order to perform administration operations, he or she is granted universal operating system rights - rights you might not want to grant in certain environments. Even if there is no malicious intent, systems can be rendered inoperable by genuine mistakes. TACF treats root in the same way as any other user when deciding whether or not to grant access to a resource.

Therefore, TACF is the perfect tool to split the universal root rights into specific job tasks (roles) which can be granted to groups of administrators according to their needs. There are many advantages for doing this, and you will have seen examples in our application lockdown modules. The administrator logs in using their own (non-root) ID and then switch user (su) to root if required by the function. The activities that administrator can perform (including the ability to `su` to root) are all determined by TACF rules.

Again, note that this is done by implementing a second layer of security to the UNIX operating system in a non-intrusive way, that is, without the need for a kernel rebuild.

To use the role-based security functions, it is best to prevent direct root logon to the system, and, instead, force the administrators to `su` to root. Doing so enables the operating system to log these actions into a log file and gives TACF the opportunity to identify the original user and grant the corresponding rights.

Unfortunately, in the case of operating systems, these role-based resource rights require the discovery and evaluation of many operating system resources, and, therefore, a deeper understanding of the operating system and its resource functions and dependencies. The AIX LDF described in the remainder of this chapter should be carefully examined and adapted to your individual needs as well as being thoroughly tested for proper operating system functions.

### 4.7.1 AIX Considerations

AIX has natively offered a form of role-based security function since version 4.2.1. Tivoli Security Management doesn’t use these capabilities because they are not available or consistent with other UNIX platforms. However, some helpful AIX role definitions can be found in /etc/security/roles to give you an idea of what additional roles might be useful in your environment.

Note that certain AIX Installations might be different depending on the installed AIX version or the installed filesets. Furthermore, the provided resource list might not match the reality due to the fact that some security-relevant files have to be created manually in order to take effect.

As mentioned above, TACF uses the login-ID the user originally logged in with to apply role-based resource rights to the user. We could try to restrict the capabilities of root and grant all the necessary rights to individual users to allow them to perform root-type functions. It is much simpler and easier to maintain a configuration where users switch to root as they need to when working on the system. Therefore, it is necessary to prevent the root user from direct system logons. This can be done using the AIX SMIT utility `smit chuser`, then select **-> root -> User can LOGIN (REMOTELY)? -> ’false’**. For systems in a physically secure environment, one might still grant local logins to root using the system console and only prevent remote login attempts.

**Note**

One can check the current user ID that TACF will base its security decisions on by invoking the `sewhoami` command.

Another important AIX system feature is the option of creating bootable system backups (mksysb). To create the corresponding file archive, the system needs to read every UX:FILE resource. This has to be kept in mind when defining default resource rights of none [N]. In our lockdown modules, we therefore defined a *SystemBackup* Role defining conditional rights to these resources. This role was granted to our sample *System_Administrators* group in order to perform these system backups.

Note that TACF itself defines three UX:FILE resources with default access of none. To change this for our backup role, we had to include these resources in our resource list for granting the required rights. Another way of granting access to these resources is to make the backup program a privileged program. We didn't test this, but the following description was found on the Tivoli support web page:

> ***Question****: mksysb fails on a box running TACF as root - cannot access "/usr/local/Tivoli/TACF/seosdb"*
>
> ***Answer****: The seosdb directory entries are protected by TACF. You can make the backup program a privileged program which should allow it to work with this directory. You will need to update a file called privpgms.init which (by default) resides in /usr/seos/etc.*
>
> *If the file does not exist, an internal list is used. When the file is created it is used instead. You should check what is currently used. Look at the startup messages for "INFO: privileged program xxx is registered". Each program listed needs a separate line in the new file. A line should then be added for the backup program. Each entry (line) should give the full path name to the program. You will then need to restart TACF and check that the backup program is in the startup messages.*

Before showing you the LDF details, there is one more AIX security feature we want to mention. AIX provides a script named /etc/securetcpip to disable certain insecure programs like tftp, rcp, rlogin and so on. You should consider running this script as a part of your security policy.

### 4.7.2 Design Considerations of the AIX LDF

Where to start is generally answered more easily in the case of locking down applications. It is more complicated in the case of operating systems.

A good starting point is to identify a basic collection of security-relevant resources to include in the lockdown definition file. Another starting point would be to consider the requirements of one or two specific roles like System Administrator or Network Administrator. You can then start assigning resources to these roles.

In our case, it turned out to be a bit of both. We started with some roles we wanted to implement and discovered many resources we wanted to assign to these roles. Then, while discovering many more security-relevant resources, the need for more roles became apparent and so on.

You should be careful not to completely lock out the root or tmesec users while distributing resource definitions to the system that are set too strong and end up preventing normal operating system functions. It proved to be helpful having a tmesec session open for test and problem determination while distributing new profiles. You might also want to deactivate the automatic TACF daemon startup in /etc/inittab while running the system in development mode.

While we normally recommend the TACF warning mode when designing and testing application lockdown modules, we preferred to work without this TACF feature in the case of the AIX lockdown to immediately notice malfunctions in the operating system. While tracing for deny-messages in the TACF log file, the operating system turned out to be very stable when denying resource access and providing the user with meaningful error messages.

### 4.7.3 TSM Resources in the AIX LDF

As already mentioned, we started our resource investigation with some security-relevant files like /etc/passwd and /etc/security/passwd of the type UX:FILE. While reviewing the AIX LDF, you will notice that we tried some relational resource access with these files due to the fact that user relevant information is stored in here.

The first two roles we tried to implement were the *Role_of_SystemAdministrator* and *Role_of_NetworkAdministrator* resulting in some more resources of type UX:FILE, including several system startup and configuration files.

This activity showed it was required of some more roles for different administration tasks as described in the next section. The corresponding resources were then easily discovered. A good source for further role-based resource investigation are the operating system man pages which usually provide a list of related files. Other sources of information are a company security policy (if available) and AIX operating system documentation.

A continuous watch of the TACF trace file completes the picture of system resource requests and, of course, the resource list.

Another group of resources of type UX:PROCESS had to be added and assigned to the roles to include all the main system processes (daemons). These daemons resulted in UX:FILE resources again enlarging our resource list.

Furthermore, some files of resource type UX:PROGRAM have to be added and can be found by searching for files with the setuid/setgid bit set.

This will complete our list of different resource types. In production environments, you could think about the need to include control for communication ports (UX:TCP) or to control the login capabilities with resources of UX:TERMINAL type.

### 4.7.4 Groups and Roles in the AIX LDF

The following information describes groups and roles in the AIX LDF:

```
GROUPS
   Everyone "Everyone" =   audit(LF,RF) ;
   Network_Administrators "Network Administrators" =   audit(LF,RF) ;
   Root_User "Root User Group" = (root)  audit(LF,RF) ;
   Secure_Users "Secure Users" =   audit(LF,RF) ;
   System_Administrators "System Administrators" =   audit(LF,RF) ;
ROLES
   Role_of_Accounting "Role of Accounting" = (System_Administrators) ;
   Role_of_Auditor "Role of Auditor" = (System_Administrators) ;
   Role_of_Everyone "Role of Everyone" =
   (System_Administrators,Everyone,Secure_Users) ;
   Role_of_HomeDirectoryAccess "Role of Home Directory Access" =
   (Secure_Users) ;
   Role_of_NetworkAdministrator "Role of Network Administrator" =
   (Network_Administrators,Root_User) ;
   Role_of_SystemAdministrator "Role of System Administrator" =
   (System_Administrators,Root_User)
   pass(UXResTypeAccess:"CONNECT,SURROGATE") ;
   Role_of_SystemBackup "Role of System Backup" =
   (System_Administrators,Root_User) pass(UXResTypeAccess:"CONNECT") ;
   Role_of_SystemResourceControl "Role of System Resource Control" =
   (System_Administrators,Root_User) ;
   Role_of_SystemShutdown "Role of System Shutdown" =
   (Root_User,System_Administrators) ;
```

```
Role_of_UserAdministration "Role of User Administration" =
(System_Administrators) ;
Role_of_root "Role of root" = (Root_User)
pass(UXResTypeAccess:"CONNECT,PROCESS") ;
```

**Role_of_Everyone**
This role is assigned to the sample group `Everyone`. `Role_of_Everyone` provides conditional resource rights to some important files like /etc/passwd and /etc/security/passwd. This list resulted from tracing normal operating system usage with the TACF trace facility and might not be complete for all implementations. Furthermore, the role defines necessary access rights to schedule cron jobs and to display a process list. This is the right place to insert additional resource rights you wish to grant to every user. In a real implementation, you would give this role to every group that required general access to this system.

**Role_of_NetworkAdministrator**
This role defines the resource access to all relevant configuration files, programs, and processes to administer the networking options of the operating system. Our sample group is named *Network_Administrator.*

**Role_of_SystemAdministrator**
This role is granting access to some configuration files and system startup resources as well as to some daemons we saw needed to be administered by the system administrator. This can easily be adapted for production environments by simply changing the assignments in the LDF file. This demonstrates the power and flexibility of this TSM resource definition meta format. This is an important role for a group such as *System_Administrators*.

Note the usage of some conditional rights for additional access to the system password files as well as to force the configuration of some important system files through the appropriate tools.

For example:

```
/etc/rc.tcpip /usr/sbin/chrctcp[F],
/etc/security/sysck.cfg /usr/sbin/chtcb[RW],
```

To provide consistency for installing software products using the AIX installation tool, we granted the following conditional right:

```
/var/adm/sw/* /usr/sbin/installp[RW],
```

In another attempt to force the administrator to use the appropriate tools for system administration tasks, we allowed the definition of `cron` jobs only through the `crontab` program:

```
/var/spool/cron/crontabs/root /usr/bin/crontab[F],
```

In our example, we have some more roles assigned to the *System_Administrators* group presented here in alphabetical order:

**Role_of_Accounting**
Provides access to major accounting information files. In live environments, you might want to grant this role to a separate function if accounting is necessary.

**Role_of_Auditor**
Grants access to system log files for auditing purposes. In live environments, you might not grant this role to the administrators group so you separate accounting functions from administrator functions.

**Role_of_Everyone**
Of course, the administrators also need basic access through conditional rights to the system password files or the user login records.

**Role_of_SystemBackup**
This role basically grants conditional rights to all resources with default rights of none [N] to the system backup facility (mksysb). Note that in cases of no access to directories, additional rights must be granted to the `/usr/bin/find` command to ensure proper operation of the system backup.

**Role_of_SystemResourceControl**
Due to the fact that some daemons in the AIX operating system are controlled through the system resource controller (SRC), we provided a separate role for control of these resources while granting access to the basic SRC control mechanisms. To provide the corresponding UX:PROCESS protection, one must include this resource definition as well as the UX:FILE expression of the process.

**Role_of_SystemShutdown**
Grants the ability to shut down the system or perform a system reboot.

**Role_of_UserAdministration**
This role provides an administrator with the necessary resource rights to perform user administration. Note that in our example,

the corresponding system files can only be accessed using conditional rights.

We included two more groups with corresponding roles. The one named *Secure_Users* with the *Role_of_HomeDirectoryAccess* gives an example of how to protect a home directory against access by any other user, even root and the administrators.

The other group named *Root_User* implements the option of still having a super user (root) outside the TACF functionality of splitting root capabilities.

Note that the *Root_User* group is granted additional resource rights using the *Role_of_NetworkAdministrator*, *Role_of_SystemAdministrator*, *Role_of_SystemResourceControl* and *Role_of_Shutdown*. In the sample LDF file, this provides full system control using the root user and should be restricted in customer environments.

# Appendix A. TSM Profile Tools

This appendix describes the `wldfexp` tool provided on the accompanying CD which will export a Tivoli Security Profile into the Lockdown Definition File (LDF) format suitable for modification for use in a lockdown module.

Also included here is a *man page* type overview of the wldfimp LDF import utility. For a more detailed description of the use of wldfimp, refer to Appendix B, “Lockdown Definition File Format” on page 105.

## A.1 The LDF Export Tool

The LDF export tool (wldfexp) reads an existing security profile and creates an LDF file that can be used to recreate the original profile. The tool does not handle TSM System Policy records, but these are not part of a typical Lockdown Module.

### A.1.1 Export Tool Usage

The LDF export tool is named wldfexp. The only mandatory parameter is the security profile name. Its usage is as follows:

```
wldfexp [flags] <profile name>
```

| | |
|---|---|
| `<profile name>` | This is the name of the profile being exported. It is also the name of the resulting LDF (profile name with LDF extension, and spaces converted into underscores). |

### A.1.2 wldfexp Flags

LDF files are best kept endpoint-type specific, and the security profile may contain data for both UNIX and Windows NT platforms. However, the export cannot handle both simultaneously. Note that other endpoints such as RACF are not supported in this version.

**-nt** Extracts NT-specific records from the profile.

**-ux** Extracts UNIX-specific records from the profile.

The following flags deal with `wldfexp` behavior.

**-f** Read the data from existing files instead of the TSM profiles. Normally `wldfexp` gets it’s information directly from the TMR using TSM, but, in some cases, it is necessary to get it from files. The files contain the output of the `wlssec` command. Using this flag without the files will not generate an error but will result in an empty LDF. You can also use this flag after the `wldfexp` has been executed with the -k flag.

| | |
|---|---|
| **-k** | `wldfexp` creates some temporary files to contain the data extracted from the profile. These are normally deleted after the tool has completed execution. Since `wlssec` can take some time, this option can make constant reruns of the tool faster. This is very useful when trying to adjust the resulting LDF (for example, when defining the variables). To do this, run the `wldfexp` once with this flag which keeps the temporary files, and then run it in the future with this flag and the -f flag. |
| **-s** | The normal ACCESS section is generated in the order the data is exported from the security profile. This can cause confusing LDF files. The -s flag makes the tool sort the ACCESS entries by resource name. |
| **-t** | Disable TCPAccess conversion to variables. The normal behavior of `wldfexp` is to take the TCPAccess statement and convert the access list into a ksh variable. This makes it easy to set up the access list; the variable just has to be set in the prolog files. |

Variable substitution options:

| | |
|---|---|
| **-v <var file>** | Load variable substitutions from a file. The format of this file is described below. <var file> is the file name to be loaded and more than one can be specified. |
| **-V <var def>** | Add a variable to the replacement list; see the information below on how to define the variable. |
| **-vd** | List the variables that have been defined; this is useful for checking the variable declaration. |

### A.1.3 Variable Substitution

Variable substitution is an important requirement when exporting Lockdown Modules for reuse. Note that there is no checking as to whether the variables are correct or make sense. Trial and error may be needed to obtain the best variables.

The substitution occurs in resource names only. A name is checked against all the variable substitutions in the order they were declared. If there is a match, the name is replaced with the variable. Variables that were declared in the LDF format (that is ${VARNAME}) are also included in the LDF file in the VAR declaration section.

Variables can be declared in a file or when invoking the export tool. The format is:

```
<variable name>=<variable value>
```

The *variable value* is used for string matching, and, when it is found, *variable name* is used to replace the original value. As substitutions are applied one at a time, subsequent substitutions for the same name result in the new substitution being used.

If declared in a file, no quotes are needed; the file is just a number of declarations. Each one must be on a line by itself with no comments or blank lines. If declared on the command invocation, quotes may be needed. Simple quotes should be used if the declaration contains a dollar sign ($) in it. Here is an example of a variable substitution file:

```
${HYPAPPHOME}=/usr/local/hypapp
${HYPAPPBIN}=${HYPAPPHOME}/bin
${HYPAPPDB}=${HYPAPPHOME}/data
${HYPAPPDUMP}=${HYPAPPHOME}/dump
${HYPAPPCONFIG}=${HYPAPPBIN}/cfg
${HYPAPPD}=${HYPAPPBIN}/hypappd
$HOSTFILE=/etc/host
```

Where we have provided a variable substitution file with an LDF, it is named with an *ldv* extension. If a Lockdown Module is exported on a regular basis, it may be useful to save and update the variable files in order to allow easy export.

## A.2 The LDF Import Utility

The LDF import utility is provided to convert LDF files into the script that will create the Lockdown Module’s security profile. Several checks are performed to make sure the script is valid. For the correct format of the LDF files, see Appendix B, “Lockdown Definition File Format” on page 105.

### A.2.1 Import Utility Usage

The conversion utility has a few options, and the basic use is:

```
wldfimp [flags] <filename>
```

| | |
|---|---|
| `<filename>` | is the name of the LDF file. The file must be a valid LDF file matching the endpoint type specified to the utility. |

### A.2.2 wldfimp Flags

The import utility deals with one kind of endpoint at a time. Specifying a Lockdown for the wrong type will cause the file to be refused by `wldfimp`. These are the endpoint type options:

| | |
|---|---|
| **-ux** | Tells the utility to treat the LDF file as a UNIX Lockdown definition. |

|  |  |
|---|---|
| **-nt** | Tells the utility to treat the LDF file as a Windows NT Lockdown definition. |

Other options:

|  |  |
|---|---|
| **-v** | This is verbose mode; show processing options and track progress. |
| **-D <vardef>** | Define a variable. These variables are defined before those of the LDF file; so, this can be used to pass site-dependent information. |
| **-o <filename>** | Write output script to a file specified by <filename>. By default, the file name is set to the Lockdown file's name with an extension of *.sh*. |
| **-p <filename>** | Use <filename> as the preface file. The default preface file is ldfc.pre from the same directory of the compiler. |

### A.2.3 Import Utility Considerations

When using variable definitions at invocation time, remember that these definitions take effect before those contained in the LDF file. If the variable is redeclared in the LDF file, a warning is issued. Variable expansion is done immediately for each -D option; so, the following command will define HOME as `/usr/local` and VARB as `/usr/local/app`:

```
wldfimp -D HOME=/usr/local -D 'VARB=${HOME}/app' sample.ldf
```

If another preface file is specified, it must ensure that these conditions are met:

- The Profile Manager target for the security profile must exist. It's name is given by the `${PRFMGR}` variable.
- The security profile must exist. The name is given by the content of the `${PROFILE}` variable.
- The security profile must be empty.

# Appendix B. Lockdown Definition File Format

This appendix provides a more formal description of the format and usage of the Lockdown Definition File (LDF) and is intended as a reference.

## B.1 LDF Component Notation

The LDF import utility reads the source file (a plain text file). Each topic below describes what types of text and markers can appear in the LDF file. The description of the layout of each section of the file starts in “LDF Section Description” on page 111.

### B.1.1 Comments

The comment symbol is used to inform the import utility that it must ignore the text that follows to the end of the line. For the LDF import utility, the comments symbol is the pound or hash sign (#). The comment may begin in any position on the line.

### B.1.2 Tokens

Tokens are words or symbols that have a special meaning in the LDF *language*. When one of them is found, the import utility first checks to see if that token is expected. The tokens usually indicate the start of a new section in the LDF. The following words are considered tokens in the LDF:

- LOCKDOWN
- VAR
- PROLOG
- GROUPS
- ROLES
- RESOURCES
- EPILOG
- ROLE
- RESOURCE
- ACCESS

The LDF import utility is case-sensitive, and the following characters are also considered tokens:

```
: ; , ( ) [ ] =
```

All the other characters will be considered part of an identifier (see below).

© Copyright IBM Corp. 1999

**Note**

Individual entries within an LDF section must be separated with a semicolon (;). Parameters within an entry are separated in the same way as the equivalent command-line options, usually with a space or a comma (,).

### B.1.3 Identifiers

In the descriptions of the LDF sections that follow, we use the term identifier, or Ident to represent a name or identifier. In the case of the LDF, almost everything except the tokens is a name. Two things can be considered a name: a string of characters that are not tokens and a quoted string that can include words that would otherwise be considered tokens. These must be quoted using the double quotation mark ("). Between the quotes, everything is part of the name. So, if you need to use a character or name that is also a token, it should be quoted. Here are some examples of strings that would be seen by the import utility as Ident:

- /usr/local/Tivoli
- $BINDIR/TACF
- "C:\Winnt\System32\User Profiles"
- C:\Winnt
- "Human Resources Role"

**Note**

The import utility has a special case for the colon token (:). Normally, the colon is considered a token and thus cannot be used inside the name without the use of quoted strings, but if the colon follows a single letter, the whole string is considered a path specification and treated as an Ident.

When declaring Groups, Roles, and Resources, it is possible to specify the name of the resource and a description for it before specifying other parameters. This is shown as two Ident separated by a space. The description is optional and should be enclosed in quotation marks. This is not mandatory. The first word is assumed to be the record name, and the second and subsequent words are assumed to be the description. Caution should be taken when declaring record names that are composed of multiple words separated by a space. A record name that contains spaces should also be enclosed in quotes. If the name is not enclosed in quotation marks, only the first word will be used as the record name, and subsequent words will become part of the record description text.

For example the entry:

```
My Group Group of People = (user1, user2)  audit(LF,RF)
```

Will result in a group being created called *My* with the text description of *Group Group of People*. In order to specify this correctly, we need quotes around the name of the group:

```
"My Group" Group of People = (user1, user2)  audit(LF,RF)
```

### B.1.4 Lists

Lists can be defined in various parts of the LDF. They are a list of identifiers separated by commas and enclosed in parentheses. Acceptable values in the list depend on where in the LDF the list is being used. The import utility notifies you when there are errors on the structure of a list. When used in GROUPS and ROLES declarations, the content of the list is subjected to variable expansion. A list will be of the form:

```
(item1, item2, item3)
```

### B.1.5 Record Properties

Each of the three Tivoli Security Management record types (Group, Role, and Resource) have a set of attributes that can optionally be defined. They govern specific aspects of the record’s behavior. All of them can be set through the GUI and CLI interface of TSM.

In the LDF, we have a data type known as Record Properties which makes the LDF capable of defining these attributes. The basic properties are handled by the normal declaration format. No FILE resource has any meaning without a name and default access permission. However, other aspects may be necessary for control (like time restrictions and auditing). It is for these optional attributes that we define the Record Property for LDF.

The actual value of the property and where it can be used depends on many aspects, but the general format is:

```
PropertyName <parameters>
```

The valid names and expected parameters are listed in Table 3. A more detailed analysis of each property type is provided after the table.

*Table 3. Record Properties*

| Name | Parameter | Example of usage | Notes |
|---|---|---|---|
| trusted | | `trusted` | Unix only |
| nottrustred | | `nottrusted` | Unix only |

| Name | Parameter | Example of usage | Notes |
|---|---|---|---|
| warning | | `warning` | Unix only |
| time | (####:####) | `time(0800:1800)` | Valid for Logon and on Unix for access to resources. |
| days | List | `days(Mon,Tue,Sat)` | as time above |
| audit | List | `audit(LA,RF)` | |
| tcpaccess | List | `tcpacces(+dover)` | Unix only |
| parent | parent role | `parent(rolename)` | Only for roles |
| pass | attribute:value | `pass(NTName:"Adm")` | |

### B.1.5.1 Trusted, Nottrusted and Warning Properties

These properties are available only in a UNIX LDF and only for resource declarations. They take no parameter and are mapped to the security CLI equivalent options as follows:

- *trusted* is mapped to attribute Trusted=T.
- *nottrusted* is mapped to attribute Trusted=F.
- *warning* is mapped to attribute Warning=T.

### B.1.5.2 Time and Days Properties

These properties are used to define Group LoginTimes and Resource AccessTimes (only available for UNIX Lockdown). If used in the group declaration, they are mapped to LoginTimes attribute. When used in a resource declaration, they are mapped to AccessTimes attribute.

If only one of the two is specified, the other is set to its default value. If neither are specified, the default values are used.

The valid values are those used by the normal TSM CLI commands. (See the description of security group attributes for wcrtsec in the product manual: *TME 10 Security Management User's Guide*.)

### B.1.5.3 Audit Property

The audit property is available in two declarations: resource and group. In the group declarations, it is mapped to the LoginAudit and ResAudit attributes of the group. On the resource declaration, it is mapped to the ResAudit attribute.

The values accepted must be one of:

| | |
|---|---|
| **LN, LS, LF, LA** | For Login auditing levels of None, Success, Failure and All respectively. |
| **RN, RS, RF, RA** | For Resource auditing levels of None, Success, Failure and All respectively. |

Login auditing can only be specified for group declarations. The same audit property statement can be used for setting up both Resource and Login auditing. Resource auditing can be set in both group and resource declarations.

#### B.1.5.4 Tcpaccess Property

This property is only available for a UNIX LDF and for the TCP resource type. It is mapped to the TCPAccess attribute of the resource record. The parameter is a list of host (or host addresses). Each host name or address must be preceded by a plus sign (+) or minus sign (-) indicating access or no access respectively. For example:

```
tcpaccess(+dover, -bath, +london)
```

Note that in many cases, the list of hosts is not available when the LDF is created. It may be interesting to use another strategy to specify the list of hosts. Instead of using the tcpaccess property, use the pass property as follows:

```
pass(TCPAccess:"$HOSTLIST")
```

The ksh variable $HOSTLIST must be set in one of the Prolog files. It must also be in the format specified for the TCPAccess attribute (See the description of security resource attributes for the `wcrtsec` command in the product manual: *TME 10 Security Management User's Guide*.)

#### B.1.5.5 Parent Property

This property is used to specify a parent role (as in the Parent attribute of the Role Record in TSM). It is therefore only valid in role declarations. The parent role must be declared within the same LDF. There cannot be a circular role parent relationship (such as where RoleA is parent of RoleB that is parent of RoleA).

The checking for the existence of the role is only done after all roles have been declared; so, the order in which they are declared here is not relevant.

#### B.1.5.6 Pass Property

This property is not mapped to any specific attribute. It is used to allow the creator of the LDF to specify other attributes that are not handled by the LDF import utility. Here is an example:

```
ROLE newrole = pass(NTGroups:"Domain Administrators");
```

The NTGroups attribute is not handled by the import utility. But, by using the pass property, the import utility passes the argument to the specific `wcrtsec` command in the resulting LDF script.

What the import utility does is the following:

```
pass(AttributeName:AttributeValue)
```

Is added to the `wcrtsec` command as an additional:

```
-s AttributeName="AttributeValue"
```

That is added to the proper `wcrtsec` command in the output script. The pass properties are placed after the ordinary properties and other fields in the LDF.

**Important**

The import utility does not do any kind of checking on either the passed attribute’s name or its value. Using the pass property with improper values or attribute names may cause the resulting script to fail.

### B.1.6 Other Parameters

Some LDF parameters do not fit in the previous categories and are described in this section. They are used in many places throughout the LDF.

#### B.1.6.1 Permission

The *Permission* parameter is used to define both default access rights and role access rights. This must be a character string enclosed in square brackets (`[]`) that represents the valid permissions for the resource. The characters used to indicate the permissions are the same used in the CLI commands. Neither commas nor spaces are allowed. Here are some examples of valid Permissions:

- [RWD] For a FILE resource type, this would be translated to the access permissions *Read* (R), *Write* (W) and *Delete* (D).
- [BMDRS] For a Windows NT SYSTEM resource type, this would be translated to the following access permissions: *Backup* (B), *Manage Domain* (MD), and *Remote Shutdown* (RS).
- [FN] would be translated as *Full Control* (F), *No Access* (N), but this is illegal; so, the import utility will not accept it.

#### B.1.6.2 ResourceName

The *ResourceName* parameter format is used to identify resources in resource declarations and access permission declarations. The LDF import utility handles two types of declarations. The first one is only the *resource name*, and the second is *resource type* and *resource name* separated by a colon. In the first, where resource type is omitted, the resource is assumed to be of the FILE resource type. Here are some examples of resource names:

- /usr/local/Tivoli/*
- FILE:/usr/local/Tivoli/* (this is equivalent to the former one)
- DIRECTORY:C:\Winnt
- FILE:"C:\Some Directory\some file.doc"
- PROCESS:${BINDIR}/oserv
- TCP:objcall
- objcall (this would be understood as FILE:objcall)

## B.2 LDF Section Description

The following text describes each of the LDF sections. Each section begins with a token that identifies the section. Some of the sections may be omitted, see the section descriptions for details. Note that the sections must appear in the following order:

- LOCKDOWN
- VAR
- PROLOG
- GROUPS
- ROLES
- RESOURCES
- ACCESS
- EPILOG

### B.2.1 LDF LOCKDOWN Section

This section is in the following form:

```
LOCKDOWN LockDownName <ident>
```

The *LockDownName* identifier defines the name of the output file and of the security profile that will be created by the script created by the LDF import utility. The name may contain spaces if it is quoted; the second *Ident* can be used to specify a comment or remark (it is ignored by the import utility). The import utility output file name will automatically have *.sh* appended to the LockDownName.

### B.2.2 LDF VAR section

Variable declaration in the LDF is straightforward:

```
VAR ident = ident; <ident = ident;>
```

Each variable has an *Ident* as a name and another for its value. Once a variable is declared in the LDF, it can be used almost anywhere else in the LDF including subsequent variable declarations by surrounding it with curly brackets and preceding it with a dollar sign as in ${VARNAME}. It is also possible to declare these LDF variables when the import utility is invoked or to use externally defined variables using the normal format for the import utility host (such as $VARNAME for UNIX or bash on Windows NT).

Variables can be used in various parts of an LDF including the prolog and epilog files. They are not expanded inside comments and properties. The following examples show how to declare variables and use them inside other declarations.

```
VAR
  TIVDIR=/usr/local/Tivoli;
  BINDIR=${TIVDIR}/bin;
```

The expansion happens at the moment a variable or name is declared. Thus, the variable BINDIR would equate to the value */usr/local/Tivoli/bin*.

**Note**

Variable names can be any valid *Ident*; so, they may include names like ~/home$A?DIR. This is a valid name, and it will be handled by the import utility, but it makes the LDF difficult to read and maintain and should be avoided.

Variable expansion also occurs in resource declarations, group declarations, and role declarations. Here is an example of a resource declaration using variables:

```
FILE:${BINDIR}/generic/* = [N];
```

It is equivalent to the following declaration (assuming the variable value is the one given in the `VAR` example above):

```
FILE:/usr/local/Tivoli/bin/generic/* = [N];
```

Variable expansion also occurs in the Prolog, Epilog and Preface files. So it is possible to *export* the variables from the LDF into these scripts. The expansion occurs when these files are processed. There are three LDF variables that are defined automatically by the import utility (if they are not defined in the LDF or through the command line). These variables are:

| | |
|---|---|
| **PROFILE** | Name of the security profile that will be created to contain the Lockdown records. The default name is the name of the Lockdown module as specified by *LockDownName* (see B.2.1, "LDF LOCKDOWN Section" on page 111). You may specify this variable to create the profile with another name. |
| **PRFMGR** | Name of the profile manager where the security profile will be created. The default value is the same as that of the PROFILE variable. |
| **REGION** | The default value is *TME 10 Security*, and it is used to indicate in which policy region the profile manager will be created. |

**Note**

Specify the names for these variables with care. The import utility is unaware of the names of existing regions, profile managers, and profiles and may generate a script that can fail to run due, for example, to a profile of the same name already existing.

### B.2.3 LDF PROLOG Section

The PROLOG and EPILOG sections have the same format. They are declared as a list of file names separated by commas:

```
PROLOG
  filename <, filename>
```

The following is an example of a valid Prolog section:

```
PROLOG script1.sh, script2.sh, script3.sh
```

The specified files are included into the script file generated by the LDF import utility. For this reason, they must be valid korn shell (ksh) scripts. The import utility does not do any kind of checking on the syntax and content of the files. The import utility only performs variable expansions and includes the lines of the script into the output script.

If PERL or csh scripts need to be executed in the prolog, a prolog ksh file can be used to call the desired scripts (like a normal call from a ksh script).

Another important item to note is that these scripts are executed as part of the import utility's output script. They are executed only on the host where the Lockdown script is executed not on the Endpoint where the Lockdown profile will be applied. In order to execute scripts on the Endpoints themselves, you need to use Tivoli Task Libraries or AEF actions which are outside the scope of the Lockdown module.

The scripts included using PROLOG should not change the REGION, PROFILE or PRFMGR variables because this may cause the remainder of the import script to fail.

### B.2.4 LDF GROUPS Section

This section of the LDF file is used to declare the security groups that will be created by the Lockdown definition as follows:

```
GROUPS
Groupname = <UserList> <RecordProps> ;
```

A GROUP section is composed of one or more group declarations. Each of these can be just a group name (which will create an empty group). We can also specify a list of users, and/or a set of properties, for example to define auditing on the security group record.

*RecordProps* is an optional set of properties (see B.1.5, “Record Properties” on page 107). *UserList* is an optional comma-separated list of user names (see B.1.4, “Lists” on page 107). If the user name refers to a Tivoli user profile it must be qualified as such and enclosed in single quotes such as:

```
GROUPS
MyAdmins = (’US_All_UP:TivUser1’ , root)
```

Here, the group members will be TivUser1, who is specified in a Tivoli user profile named US_All_UP, and the user named root.

Variable expansion occurs in both the name of the group and the user names but not for the record properties.

### B.2.5 LDF ROLES Section

This section of the LDF file is used to declare the security roles that will be created by the Lockdown definition script. This should not be confused with the ROLE entry used to update access to resources in the ACCESS section (see B.2.7, “LDF ACCESS Section” on page 115). The ROLES section uses the format:

```
ROLES
Rolename = <GroupList> <Recordprops> ;
```

The ROLES section is composed of one or more role declarations. Each of these can be formed by either the role name or the role name and a list of groups and/or role properties.

*GroupList* is an optional list of group names (see B.1.4, “Lists” on page 107). The names declared in the list must have been declared in the Group Declaration section of the LDF. If it is necessary to set up different groups than those declared in the group section; this should be done using the pass property (see B.1.5.6, “Pass Property” on page 109) and/or an Epilog file (see B.2.8, “LDF EPILOG Section” on page 117).

*RecordProps* is an optional set of Role properties (see B.1.5, “Record Properties” on page 107).

Variable expansion occurs in the role name and in the group list but not in the record properties.

### B.2.6 LDF RESOURCES Section

This section is used to declare all the resources that will be handled by the Lockdown definition. It has the form:

```
RESOURCES
ResourceName <ident> = Permission <RecordProps> ;
```

Only resources declared in this section may be used in the role ACCESS section (see B.2.7, “LDF ACCESS Section” on page 115).

The RESOURCES section is composed of one or more resource declarations. Each of these is composed of a *ResourceName* and a *Permission* (see B.1.6, “Other Parameters” on page 110). Optionally, there can also be a set of *RecordProps* (see B.1.5, “Record Properties” on page 107). The *ResourceName* can be followed by an optional identifier that can be used as a comment.

Permission defines the Default Access Permission for the resource. In this section, it is not possible to add conditional access permissions. This can be done using the ACCESS section which is described next.

### B.2.7 LDF ACCESS Section

This is the most important part of the LDF file. This section is used to define which access permissions are granted to a specific role and on certain resources attached to that role. It has the form:

```
ACCESS
ROLE Ident <role access parameters>
```

```
RESOURCE restype <resource access parameters>
```

A grant can be done in two ways: Either by Role or by Resource. The option is identified by the ROLE and RESOURCE tokens respectively.

When the grant is done to a role, we can list all the resources we wish to define access for in that role. This will be the preferred method if we have a small number of roles to define. The statement should be in this format:

```
ROLE RoleName =
  FILE:/filename [R],
  PROCESS:/processname [A];
```

After the equal sign, all you have is a list of comma-separated resources and the access granted to the role named after the ROLE token.

The other way of specifying access permissions is by listing which roles have what access to a given resource. This form is better when you wish to define the access for many roles to similar resources and where the parent/child relationship is not sufficient to meet your needs. To achieve this, use the following format:

```
RESOURCE PROCESS:/processname =
  RoleNameA [A],
  RoleNameB [A],
  RoleNameC [N];
```

The resource on which the permissions will be given is the one immediately after the RESOURCE token. After the equal sign is a list of comma-separated role names and the access each one has to the specified resource.

In both formats, it is possible to grant conditional access to resources. In order to specify a conditional access to a resource, the *AccessDefinition* must follow the first option (with *ViaAccess* and *Permission*). The format for conditional access in the LDF is:

```
ROLE RoleName = /etc/passwd /usr/bin/passwd[RW];
RESOURCE /etc/passwd = RoleName /usr/bin/passwd[RW];
```

Both are equivalent, and will generate the following on the RoleName role:

```
-s UXTMEResAccess="/etc/passwd perms(R,W) via(PROGRAM:/usr/bin/passwd)"
```

**Note**

At the time of writing, the import utility only supported the ROLE declaration of access and not the RESOURCE declaration. See 1.1, “Obtaining Lockdown Updates” on page 5 for information about obtaining updates to the LDF files and the utilities.

### B.2.8 LDF EPILOG Section

The EPILOG and PROLOG sections have the same format. They are declared as a list of file names separated by commas:

```
EPILOG
  filename <, filename>
```

The following is an example of a valid EPILOG section:

```
EPILOG scriptx.sh, scripty.sh, scriptz.sh
```

The specified files are included into the script file generated by the LDF import utility. For this reason, they must be valid korn shell (ksh) scripts. The import utility does not do any kind of checking on the syntax and content of the files. The import utility only performs variable expansions and includes the lines of the script into the output script.

If PERL or csh scripts need to be executed in the epilog, an epilog ksh file can be used to call the desired scripts (like a normal call from a ksh script).

Another important item to note is that these scripts are executed as part of the import utility’s output script. They are executed only on the host where the Lockdown script is executed not on the Endpoint where the Lockdown profile will be applied. In order to execute scripts on the Endpoints themselves, you need to use Tivoli Task Libraries or AEF actions which are outside the scope of the Lockdown module.

The scripts included using EPILOG should not change the REGION, PROFILE or PRFMGR variables because this may cause the remainder of the LDF import script to fail.

# Appendix C. Special Notices

This publication is intended to help those implementing Tivoli Security Management achieve a working environment more rapidly. The information in this publication is not intended as the specification of any programming interfaces that are provided by the Tivoli Security products. See the PUBLICATIONS section of the IBM Programming Announcement for Tivoli Security Management for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have

been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

| | |
|---|---|
| AS/400 | AT |
| BookManager | CICS |
| CT | DB2 |
| eNetwork | Home Director |
| IBM ® | Language Environment |
| OS/390 | OS/400 |
| PROFS | RACF |
| RS/6000 | S/390 |
| SP | System/390 |
| XT | 400 |

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. (For further information, see http//www.setco.org/aboutmark.html).

Other company, product, and service names may be trademarks or service marks of others.

# Appendix D.  Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## D.1  International Technical Support Organization Publications

For information on ordering these ITSO publications, see “How to Get ITSO Redbooks” on page 125.

- *Tivoli Security Management Design Guide*, SG24-5101
- *Managing Access from Desktop to Datacenter: Introducing TME 10 Security Management*, SG24-2021
- *Tivoli Enterprise Internals and Problem Determination*, SG24-2034-01

## D.2  Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. **Order a subscription** and receive updates 2-4 times a year.

| **CD-ROM Title** | **Collection Kit Number** |
|---|---|
| Tivoli Redbooks Collection | SK2T-8044 |
| System/390 Redbooks Collection | SK2T-2177 |
| Networking and Systems Management Redbooks Collection | SK2T-6022 |
| Transaction Processing and Data Management Redbook | SK2T-8038 |
| Lotus Redbooks Collection | SK2T-8039 |
| AS/400 Redbooks Collection | SK2T-2849 |
| RS/6000 Redbooks Collection (HTML, BkMgr) | SK2T-8040 |
| RS/6000 Redbooks Collection (PostScript) | SK2T-8041 |
| RS/6000 Redbooks Collection (PDF Format) | SK2T-8043 |
| Application Development Redbooks Collection | SK2T-8037 |

## D.3  Other Publications

The following Microsoft publications mentioned in this redbook are Microsoft Knowledge Base documents and can be found at the following Web site:

`http://www.microsoft.com/NTServer/nts/techdetails/overview/WpGlobal.asp`

- *Microsoft Windows NT - Securing Windows NT Installation*, October 23,1997, Microsoft Corporation

- *How to Disable LM Authentication on Windows NT*, Microsoft Knowledge Base article Q147706.

The following Trusted Systems Services document mentioned in this redbook can be found at the following Website:
`http://www.trustedsystems.com/NSAGuide.htm`

- *Windows NT Security Guidelines*, A study for NSA Research by Trusted Systems Services

The following Tivoli Publications are product documentation, which can only be obtained by purchasing the associated Tivoli product:

- *TME 10 Security Management User Guide*
- *Security Management Reference Manual for TACF*

## How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders via e-mail including information from the redbooks fax order form to:

|  | **e-mail address** |
|---|---|
| In United States | usib6fpl@ibmmail.com |
| Outside North America | Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/` |

- **Telephone Orders**

|  |  |
|---|---|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/` |

- **Fax Orders**

|  |  |
|---|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-403-267-4455 |
| Outside North America | Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/` |

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at `http://www.redbooks.ibm.com/` and for IBM employees at `http://w3.itso.ibm.com/`.

**IBM Intranet for Employees**

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may also view redbook. residency, and workshop announcements at `http://inews.ibm.com/`.

## IBM Redbook Fax Order Form

**Please send me the following:**

| Title | Order Number | Quantity |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| First name | Last name | |
| Company | | |
| Address | | |
| City | Postal code | Country |
| Telephone number | Telefax number | VAT number |

☐ Invoice to customer number ____________________

☐ Credit card number ____________________

| | | |
|---|---|---|
| Credit card expiration date | Card issued to | Signature |

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

# List of Abbreviations

| | |
|---|---|
| ***ACC*** | AutoPack Control Center |
| ***ACE*** | Access Control Entries |
| ***ACF*** | Adapter Configuration Facility |
| ***ACL*** | Access Control List or Access List (RACF) |
| ***ACP*** | (logfile) Adapter Configuration Profile |
| ***ADE*** | Application Development Environment |
| ***ADSTAR*** | ADSTAR Distributed Storage Manager |
| ***AEF*** | Tivoli 10 Application Extension Facility |
| ***AIX*** | Advanced Interactive eXecutive (IBM UNIX) |
| ***ALI*** | Authentication, Location, and Inheritance |
| ***AMS*** | Application Management Specification |
| ***ANSI*** | American National Standards Institute |
| ***APAR*** | Authorized Program Analysis Report |
| ***APF*** | Authorized Program Facility |
| ***API*** | Application Programming Interface |
| ***ARP*** | Access Resolution Protocol |
| ***AS/400*** | Application System/400 |
| ***BARC*** | Before, After, Remove, and Configuration (script) |
| ***BAROC*** | Basic Recorder of Objects in C |
| ***BDC*** | Backup Domain Controller |
| ***BDT*** | Bulk Data Transfer |
| ***BO*** | Behavior Object |
| ***BOA*** | Basic Object Adapter |
| ***CCMS*** | Configuration and Change Management System |
| ***CBPDO*** | Custom Built Product Delivery Offering |
| ***CCMS*** | Configuration and Change Management System |
| ***CDS*** | Cell Directory Service (DCE) |
| ***CICS*** | Customer Information Control System |
| ***CIM*** | Configuration Information Manager |
| ***CIO*** | Chief Information Officer |
| ***CLI*** | Command Line Interface |
| ***CORBA*** | Common Object Request Broker Architecture |
| ***CSNW*** | Client Services for NetWare |
| ***CT-LIB*** | Client-Library (Sybase) |
| ***DAC*** | Discretionary Access Control |
| ***DAP*** | Directory Access Protocol |
| ***DB*** | Database |
| ***DB2*** | IBM DataBase 2 |

© Copyright IBM Corp. 1999

| | |
|---|---|
| ***DBA*** | Database Administrator |
| ***DCE*** | Distributed Computing Environment |
| ***DES*** | Data Encryption Standard |
| ***DFS*** | Distributed File System |
| ***DFW*** | Desktop for Windows |
| ***DHCP*** | Dynamic Host Configuration Protocol |
| ***DII*** | Dynamic Invocation Interface |
| ***DLL*** | Dynamic Link Library |
| ***DM*** | Domain Manager |
| ***DMTF*** | Desktop Management Task Force |
| ***DN*** | Distinguished Name |
| ***DNS*** | Domain Name System/Server |
| ***DO*** | Display Object |
| ***DPO*** | Default Policy Object |
| ***DRM*** | Default Routing Manager |
| ***DSA*** | Digital Signature Algorithm |
| ***DSB*** | Directory Service Broker (DASCOM) |
| ***DSL*** | Dialog Specification Language |
| ***DTS*** | Distributed Time Service (DCE) |
| ***EDI*** | Electronic Data Interchange |
| ***EGID*** | Effective Group ID |
| ***EHLLAPI*** | Enhanced High Level Language APIS |
| ***EIF*** | Event Integration Facility |
| ***EP*** | Endpoint |
| ***ERA*** | Extended Registry Attributes |
| ***ES1*** | Enhanced Scalability Release 1 |
| ***ESM*** | Enterprise Server Management |
| ***FAT*** | File Allocation Table |
| ***FQHN*** | Fully Qualified Host Name |
| ***FSFI*** | Free Space Fragmentation Index |
| ***FSP*** | File Security Packet |
| ***FTP*** | File Transfer Protocol |
| ***GCOS*** | GE Computer Operating System field |
| ***GDA*** | Global Directory Agent |
| ***GEM*** | Global Enterprise Manager |
| ***GID*** | Group Identifier |
| ***GRE*** | Generic Routing Encapsulation |
| ***GSNW*** | Gateway Services for NetWare |
| ***GSO*** | (IBM or Tivoli) Global Sign-On |
| ***GSSAPI*** | Generic Security Service API |
| ***GTF*** | Generalized Trace Facility |
| ***GUI*** | Graphical User Interface |
| ***HACMP*** | High Availability Cluster Multi-Processing |
| ***HFS*** | Hierarchical File System |
| ***HR*** | Human Resources |
| ***HTTP*** | Hypertext Transport Protocol |

| | |
|---|---|
| ***IANA*** | Internet Assigned Number Authority |
| ***IBM*** | International Business Machines Corporation |
| ***IDL*** | Interface Definition Language |
| ***IOM*** | Inter Object Messaging |
| ***IPC*** | Inter-Process Communication |
| ***IPX/SPX*** | Internet Packet eXchange/Sequenced Packet eXchange |
| ***IRF*** | Inherited Rights Filter |
| ***IT*** | Information Technology |
| ***ITSO*** | International Technical Support Organization |
| ***JCL*** | Job Control Language |
| ***JES*** | Job Entry Subsystem |
| ***LAN*** | Local Area Network |
| ***LC*** | Logon Coordinator |
| ***LCF*** | Lightweight Client Framework - Now known as Tivoli Management Agent |
| ***LDAP*** | Lightweight Directory Access Protocol |
| ***LE*** | Language Environment |
| ***LQM*** | Local Queue Manager |
| ***LSA*** | Local Security Authority |
| ***LSF*** | Logon Script File |
| ***LUID*** | Login User ID |
| ***MAC*** | Message Authentication Code |
| ***MAC*** | Mandatory Access Control |
| ***MCSL*** | Monitoring Capabilities Subscription Language |
| ***MDist*** | Multiplexed Distribution |
| ***MIF*** | Management Information File |
| ***MVS*** | Multiple Virtual Storage |
| ***NAC*** | Network Application Consortium |
| ***NAT*** | Network Address Translation |
| ***NDS*** | Novell Directory Services |
| ***NetBEUI*** | NetBios Extended User Interface |
| ***NetBIOS*** | Network Basic Input Output System |
| ***NFS*** | Network File System |
| ***NIS*** | Network Information System |
| ***NLM*** | Netware Loadable Module |
| ***NLS*** | National Language Support |
| ***NQM*** | Network Queue Manager |
| ***NTFS*** | NT File System |
| ***NWMS*** | NetWare Managed Site |
| ***OCI*** | Oracle Call Interface |
| ***ODBC*** | Open Database Connectivity |
| ***OID*** | Object Identifier |
| ***OMG*** | Object Management Group |
| ***OOB*** | Out of Band |
| ***ORB*** | Object Request Broker |
| ***PAT*** | Port Address Translation |
| ***PCMN*** | PC Managed Node |
| ***PCOMM*** | (IBM eNetwork) Personal Communications (for Windows NT and 95) |

| | |
|---|---|
| ***PDC*** | Primary Domain Controller |
| ***PD/PSI*** | Problem Determination/Problem Source Isolation |
| ***PIN*** | Personal Identification Number |
| ***PKM*** | Personal Key Manager |
| ***PO*** | Prototype Object |
| ***PTF*** | Program Template File |
| ***PTF*** | Program Temporary Fix |
| ***PROFS*** | Professional Office System |
| ***RACF*** | Remote Access Control Facility |
| ***RAM*** | Random Access Memory |
| ***RCS*** | Revision Control System |
| ***RDBMS*** | Relational Database Management System |
| ***REXX*** | Restructured Extended Executor Language |
| ***RFC*** | Request For Comments |
| ***RGID*** | Real Group ID |
| ***RID*** | Real User ID |
| ***RIM*** | RDBMS Interface Module |
| ***RPC*** | Remote Procedure Call |
| ***RRSF*** | RACF Remote Sharing Facility |
| ***SAF*** | System Authorization Facility |
| ***SAM*** | Security Account Manager |
| ***SCD*** | Spoolmate Configuration Database |
| ***SDSF*** | Spool Display and Search Facility |
| ***SeOS*** | Security Operating System |
| ***SHS*** | Secure Hash Standard |
| ***SID*** | Security Identifier |
| ***SIS*** | Software Installation Service |
| ***SLIP*** | Serial Line Internet Protocol |
| ***SMB*** | Server Message Block |
| ***SMF*** | System Management Facility |
| ***SMIT*** | Systems Management Interface Tool (AIX) |
| ***SMP/E*** | System Modification Program Extended |
| ***SNMP*** | Simple Network Management Protocol |
| ***SRM*** | Security Reference Monitor |
| ***SSO*** | Single Sign-On |
| ***SVC*** | System V Communication |
| ***TACF*** | Tivoli Access Control Facility |
| ***TAP*** | Tivoli Authentication Package |
| ***TCB*** | Trusted Computing Base |
| ***TCL*** | Tool Command Language |
| ***TCP/IP*** | Transport Control Protocol/Internet Protocol |
| ***TEC*** | Tivoli Enterprise Console |
| ***TEIDL*** | Tivoli-Extended IDL |
| ***TIR*** | Tivoli Information Router |

| | |
|---|---|
| ***TLI*** | Transport Layer Interface |
| ***TLL*** | Task Library Language |
| ***TMA*** | Tivoli Management Agent |
| ***TME*** | Tivoli Management Environment (now known as Tivoli Enterprise) |
| ***TMF*** | Tivoli Management Framework |
| ***TMR*** | Tivoli Management Region or TME 10 Management Region |
| ***TNR*** | Tivoli Name Registry |
| ***TNWR*** | TME 10 NetWare Repeater |
| ***TRAA*** | Tivoli Remote Access Account |
| ***TRIP*** | Used to Refer to the Tivoli Remote Execution Service |
| ***TSM*** | Tivoli Security Management |
| ***TSO*** | Time Sharing Option |
| ***TSO/E*** | TSO Extensions |
| ***TP*** | Transaction Program |
| ***TUA*** | Tivoli User Administration |
| ***UACC*** | Universal Access Authority |
| ***UCT*** | Universal Coordinated Time |
| ***UED*** | Unison Enterprise Database |
| ***UID*** | User Identifier |
| ***URL*** | Uniform Resource Locator |
| ***VM*** | Virtual Machine |
| ***VPO*** | Validation Policy Object |
| ***VPN*** | Virtual Private Network |
| ***WAN*** | Wide Area Network |
| ***WINS*** | Windows Internet Naming Service |
| ***WTO*** | Write To Operator |
| ***XBO*** | Extended Behavior Object |
| ***XPG/4*** | X/Open Portability Guide Issue 4 |
| ***XSSO*** | X/Open Single Sign-On |

# Index

# ITSO Redbook Evaluation

Securing Applications with Tivoli Security Management Lockdown Modules
SG24-5140-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
**_ Customer   _ Business Partner   _ Solution Developer   _ IBM employee**
**_ None of the above**

**Please rate your overall satisfaction** with this book using the scale:
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

Overall Satisfaction __________

**Please answer the following questions:**

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

______________________________________________________________

______________________________________________________________

______________________________________________________________

______________________________________________________________

What other redbooks would you like to see published?

______________________________________________________________

______________________________________________________________

______________________________________________________________

**Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)**

______________________________________________________________

______________________________________________________________

______________________________________________________________

______________________________________________________________

© Copyright IBM Corp. 1999

SG24-5140-00
Printed in the U.S.A.

Securing Applications with Tivoli Security Management Lockdown Modules

SG24-5140-00