4-10

SANDIA REPORT

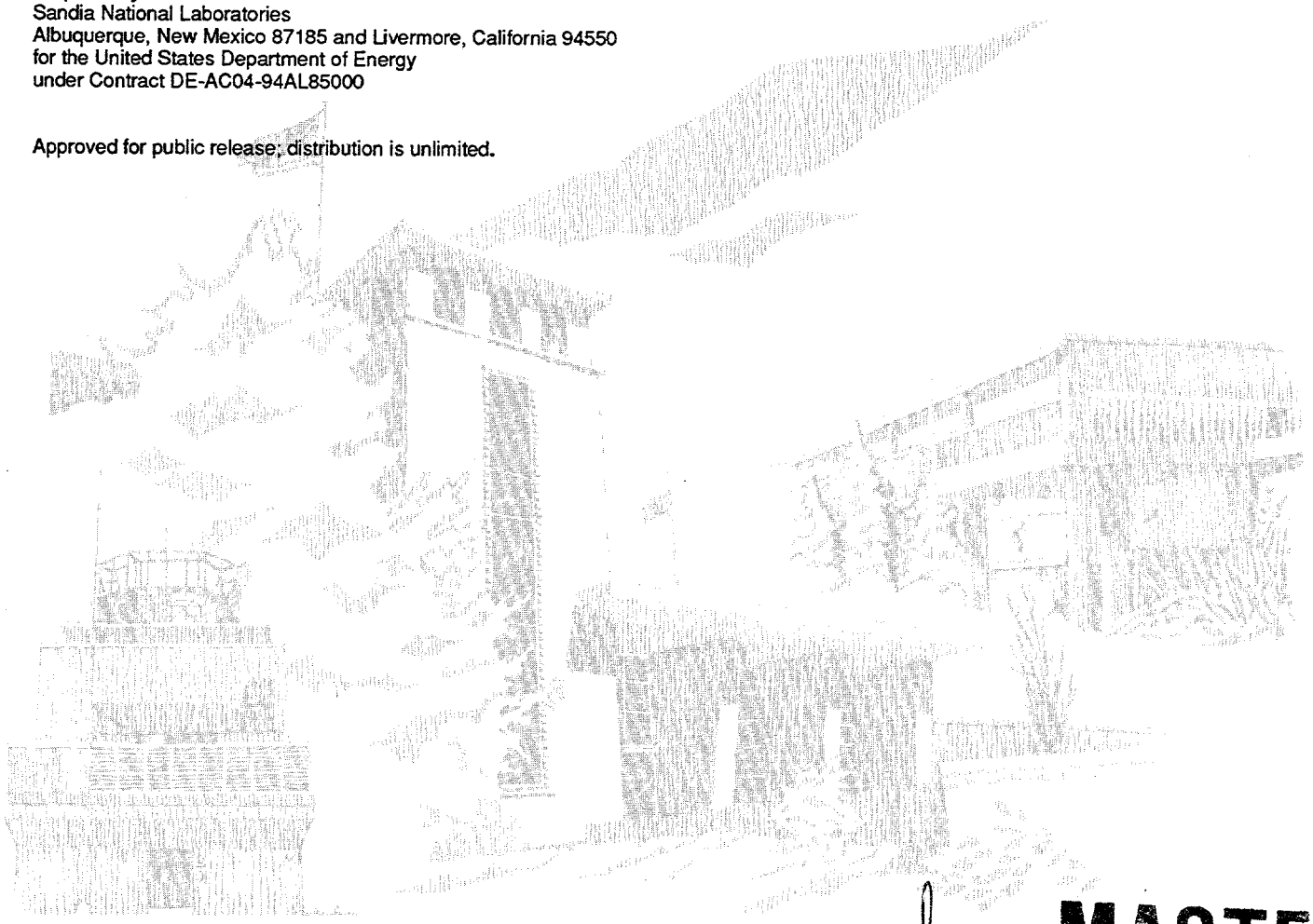
SAND96–0732 • UC–705
Unlimited Release
Printed March 1996

The Technology Information Environment with Industry™ System Description

Richard Detry, Glenn Machin

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

Approved for public release; distribution is unlimited.



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A08
Microfiche copy: A01

SAND96-0732
Unlimited Release
Printed March 1996

Distribution
Category UC-705

The Technology Information Environment with Industry™ System Description

Richard Detry and Glenn Machin
Scientific Computing Systems Department
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

The Technology Information Environment with Industry (TIE-In™) provides users with controlled access to distributed laboratory resources that are packaged in intelligent user interfaces. These interfaces help users access resources without requiring the user to have technical or computer expertise. TIE-In utilizes existing, proven technologies such as the Kerberos authentication system, X-Windows, and UNIX sockets. A Front End System (FES) authenticates users and allows them to register for resources and subsequently access them. The FES also stores status and accounting information, and provides an automated method for the resource owners to recover costs from users. The resources available through TIE-In are typically laboratory-developed applications that are used to help design, analyze, and test components in the nation's nuclear stockpile. Many of these applications can also be used by U.S. companies for non-weapons-related work. TIE-In allows these industry partners to obtain laboratory-developed technical solutions without requiring them to duplicate the technical resources (people, hardware, and software) at Sandia.

Intentionally Left Blank

Contents

Figures

Tables

Acronyms

AcctM - Accounting Manager
ADaPT - Advanced Design and Production Technologies
API - Application Programming Interface
ASCI - Accelerated SuperComputing Initiative
CFS - Common File System
CORBA - Common Object Request Broker Architecture
CSS - Common Services System
DCE- Distributed Computing Environment
DFS - Distributed File System
DOE - Department of Energy
EON - External Open Network
EVE - Enterprise Viewing Environment
FES - Front End System
GW - X11 Gateway
KAS - Kerberos Authentication Service
ICE - Interactive Collaborative Environment
IP - Internet Protocol
IRN - Internal Restricted Network
ISN - Internal Secure Network
KDC - Key Distribution Center (kerberos)
MFES - Master Front End System
NFS - Network File System
NTP - Network Time Protocol
OSF - Open Software Foundation
PC - personal computer, IBM compatible and Macintosh
PPP - Point-to-Point Protocol
RPC - remote procedure call
RRM - Resource Request Manager
SCIS - Service Center Information System
SIDS - SecurID server
SLIP - Serial Line Internet Protocol
SM - Session Manager
SME - small or medium sized enterprise
TAS - TIE-In Application Server
TCP - Transport Control Protocol
TGT - Ticket-Granting Ticket (kerberos)
TIE-In - Technology Information Environment with Industry
TPM - TIE-In Profile Manager
UDP - User Datagram Protocol
WFO - Work for Others
WS - workstation
X11GW - X11 Gateway

The Technology Information Environment with Industry System Description

Introduction

The Technology Information Environment with Industry (TIE-In) provides a new mechanism for working with a distributed set of users: remote electronic access to packaged technical solutions. This mechanism focuses on providing the nonexpert with guided solutions embedded in intelligent user interfaces, while minimizing the investment required to utilize these technologies. Technical solutions available through TIE-In include such capabilities as computational simulation, modeling and design, and development and testing facilities. These solutions can benefit not only internal Sandia employees, but the manufacturing, energy, aerospace, electronics, and automotive industries as well.

By utilizing modern information and computer technologies, TIE-In employs existing capabilities and expertise to provide practical and timely outreach to its customers. This outreach effort provides users with an on-line, on-demand service, which offers proven solutions to a wide variety of problems in a practical setting. The emphasis is on providing users with solutions using technologies at the national laboratories without requiring that users have a lot of technical expertise or high-performance computers. These technologies are packaged with intelligent user interfaces and include education and training materials, on-line help documents, expert systems and databases, guided software tools, and distributed computing resources.

The TIE-In system is based on a number of key elements: packaging of applications with intelligent user interfaces, controlled and secured on-line access to distributed laboratory application servers, and use of established graphics protocols for multiplatform compatibility. Intelligent user interfaces make it easier for lab-developed applications to be used by more than just the application developers. In general, a scientist or engineer should be able to use these technical applications without having to be a UNIX expert or spending months learning the idiosyncrasies of a piece of technical software. This packaging is especially important since TIE-In allows users to remotely access a wide variety of different resources.

TIE-In uses an integrated system to perform user authentication and to control access to lab applications and user files. By allowing users to run applications on computers that are configured and maintained at the labs, a broader range of applications can be made available to users. In contrast to traditional methods for distributing software, TIE-In avoids many of the configuration and software support issues that arise when users try to

run software on multiple platforms. This makes it practical to share packaged applications with users without having to make a large commitment to port the application to a broad range of potential user platforms. In addition, users gain the benefits of using applications that are upgraded and maintained by the scientists and engineers that developed and maintain the technology.

TIE-In was initially developed to on reduce barriers to working with small-to-medium-sized enterprises (SME) to access a wide range of Sandia-developed applications. Since its initial implementation in June 1994, it has been used by eight universities, six industrial users and eight government users. Of these, six have been paying users with a total user fee collection of over $50,000. While these external usage numbers are still low, the TIE-In system is being heavily used by internal Sandia users to access a variety of corporate and scientific applications. TIE-In is also being used as the authentication mechanism for users from the other DOE laboratories to access Sandia's classified resources via SecureNet.

The purpose of this report is to describe the main components of the TIE-In system.

TIE-In Technical Overview

This section provides a general overview of the TIE-In system infrastructure. This includes a discussion of the communication network, instructions on how to access TIE-In, and brief descriptions of the processes that comprise the TIE-In system.

The TIE-In system infrastructure is composed of four key components: the Kerberos Authentication Service (KAS), the Front End System (FES), the Common Services System (CSS), and the TIE-In Application Server (TAS). Figure 1 shows the relationship between the components of the infrastructure and the underlying network.

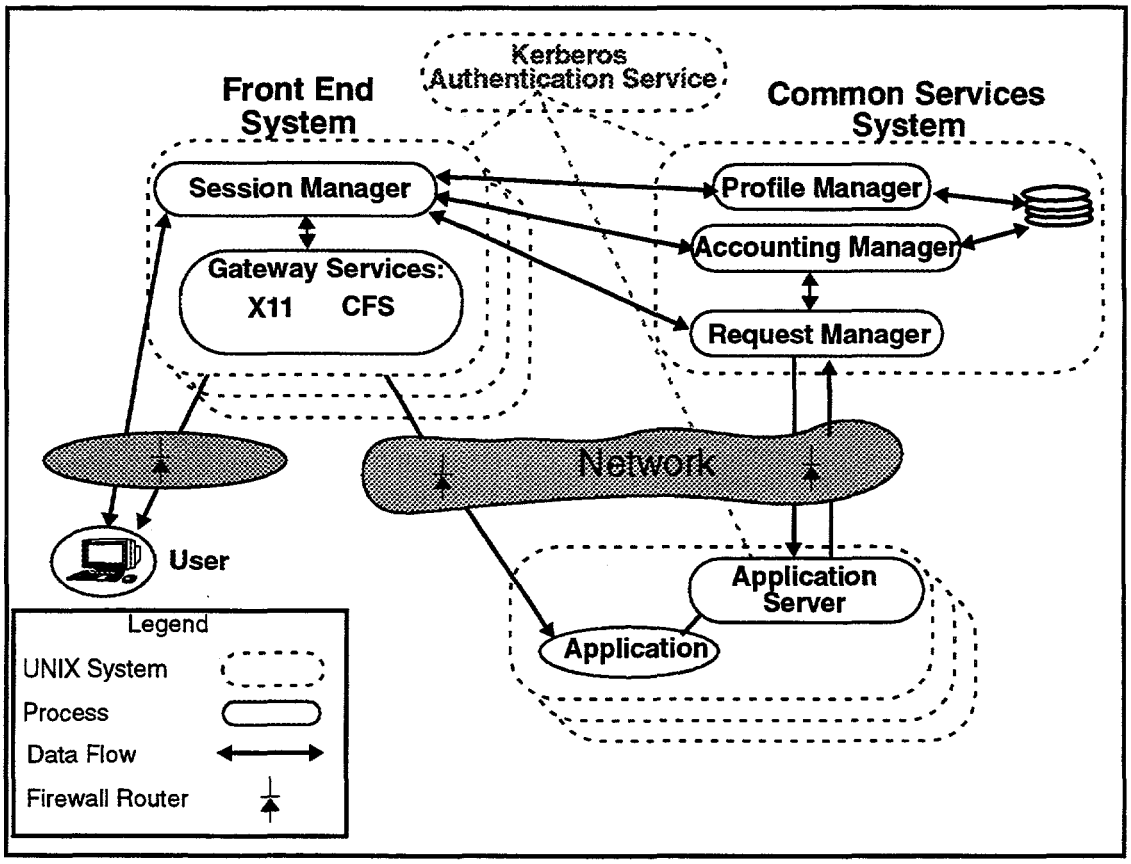


Figure 1. Schematic diagram of the TIE-In system infrastructure.

Kerberos Authentication Service

Authentication is the key to the security of TIE-In. TIE-In not only authenticates the user at the Front End System, but authenticates all interprocess communication as well. This is accomplished through the use of Kerberos tickets. Kerberos is an authentication scheme developed by MIT for project Athena.[1] It uses a trusted third-party authentication methodology utilizing private keys and has been used in Sandia's Internal Secure Network as a network-authentication service for a number of years. Each user and application registered with TIE-In has a corresponding principal entry in the Kerberos Authentication Service (KAS). The KAS issues tickets, which when presented to the server, authenticate the user and the system that issued the request.

Additional authentication of users is available through the KAS by using one-time password devices such as the Security Dynamics SecurID card. When users are further authenticated with such a device, the KAS notes this in the Ticket Granting Ticket (TGT) it issues. This information cannot be modified by any process except the KAS since the TGT is encrypted in a key known only to the KAS. This same information is contained within the tickets presented to the servers, thus allowing servers to make decisions based upon the level at which the user was authenticated.

For more information about Kerberos, see "Authentication" in the "TIE-In Security" section.

Front End System

The Front End System provides the initial user interface to TIE-In applications. This interface is provided through the Session Manager (SM), an OSF (Open Software Foundation) Motif graphical user interface. The Session Manager performs user authentication and initiates user application requests to the Request Manager, located on the Common Services System. The Session Manager is also the user's interface to the Accounting Manager and the Profile Manager, also located on the Common Services System. The Session Manager provides a wide range of administration tools for the TIE-In system and application administrators.

Gateway Service

A key service provided by the Front End System is the Gateway Service. The Gateway Service provides seamless access between applications and the user's workstation or desktop system.

X11 Gateway

The X11 Gateway (X11GW) provides seamless access from all applications to the user's workstation or PC. Applications may reside anywhere on the network and may move from system to system.

X11 servers typically require users to specify the system that will be allowed access by way of the *xhost* command or some similar mechanism, or open up access to all X11 clients. This requirement would force all users to know or be notified of where each application resides. To overcome this, the Front End System establishes an X11 gateway for the user. All applications send X11 data to the gateway, which then forwards it to the user's X11 server. With the gateway, users only need to allow X11 access to the Front End System. Since there is a slight decrease in performance associated with using the gateway, users may override this feature by disabling it through the X11 Security button within their main TIE-In window.

TIE-In also offers the user the ability to require notification of X11 access. When this option is used, a notification window will appear on the user's display prior to any connections to the user's X server by any of the TIE-In applications. The notification window asks the user if he wishes to allow the access. By default this feature is turned off, since it can be confusing to users.

The X11 gateway service incorporates the Interactive Collaborative Environment (ICE). With ICE, users may specify X displays with which they wish to share the application. This allows for collaborative design and interactive assistance with experts on TIE-In applications.

Common File System

TIE-In's Common File System (CFS) provides distributed file service access between applications and a remote file system established for the user. This allows all applications to read from and write to the user's home directory. For a detailed discussion about the CFS, see the "TIE-In Common File System" section.

Common Services System

The Common Services System maintains all user and application information, and collects accounting and usage information. All requests to Application Servers go through the Common Services System. Three processes comprise the Common Services System: the Request Manager, the Profile Manager, and the Accounting Manager.

The Request Manager is the intermediary between the Session Manager and all of the Application Server processes. The primary function of the Request Manager is to relay requests, status, and accounting information between the Session Manager, the Application

Servers, and the Accounting Manager, while monitoring the states of active tasks. The Request Manager takes Session Manager requests, places an identifier on them, and passes them to the appropriate Application Server. This process also receives state and accounting information from the Application Servers and passes it on to the Accounting Manager. When active tasks are complete, the Request Manager notifies the Gateway Service to close down any gateways established for that request. Occasionally, if status and accounting information has not recently been received for an active task, the Request Manager will query the appropriate Application Server for updated status information.

The Profile Manager is the agent between the Session and Resource managers and the TIE-In database. Its primary function is to write, update, and read all user and application information to and from the database. All users making queries or updates are checked to ensure that they have the permission to do so. Users are only allowed to modify certain fields in their own user record. Similarly, application administrators are only allowed to modify the records of applications for which they are the administrator. TIE-In administrators can modify any information contained in the database.

The Accounting Manager maintains the state of application tasks, as well as accounting information that may have been received by the Request Manager from the Application Servers. The Accounting Manager is available to the users for inquiries concerning active and completed tasks, as well as accumulated charges.

Application Server

The TIE-In Application Server is the process that runs on the machine serving the application. It authenticates and validates incoming requests, performs the authorization when users wish to register for or run an application, executes and monitors the application, and returns status and accounting information back to the Request Manager. Local system administrators have final control over who has access to an application. Through a local configuration file, the Application Server will execute authorization and account setup scripts, provided by the local administrators when users register for, or request to run, an application. Local administrators may also elect to have the application run under a captive account, or with a totally different root filesystem than other users of the system. A single Application Server can handle many applications.

Multiple Organizational Realms

TIE-In can be scaled into multiple organizational realms. Each realm is responsible for administration of its own Front End and Kerberos Authentication Systems. Users receive authentication from their local realm, but they may still access applications in different realms. This type of configuration allows organizations to share resources in a limited and controlled fashion. The Common Services System is shared by all realms, which provides information about users and applications in all realms. The Profile Manager will ensure

uniqueness of usernames and userids, which can be an area of concern when resources are shared between organizations.

System Diagram

Figure 2 is a diagram of the TIE-In system, showing the communication between the main TIE-In processes. In this diagram the Front End System, the Common Services System, and the Kerberos Authentication Service are on the same workstation. This configuration is referred to as the Master Front End System (MFES). In the current implementation of TIE-In, the MFES resides on a Hewlett Packard 735 workstation named *tiein.sandia.gov*.

In the diagram, the Resource Request Manager (RRM), TIE-In Profile Manager (TPM), and Accounting Manager (AcctM) comprise the Common Services System (CSS), while the Front End System (FES) consists of the Session Manager (SM) and the gateways. The KAS consists of the Key Distribution Center (Krb5kdc) and the administration daemon (kadmind). Although it is not shown on the diagram, the RRM, TPM, AcctM, and TAS all communicate with the KAS.

Figure 2 shows the messages that are passed between the main TIE-In processes. In each case, the client process sends a request to the server process and receives a reply. The formats of all the messages are given in Appendix A: Message Formats. The messages are also discussed in more detail in later sections describing the main TIE-In processes.

Communication Network

TIE-In is designed to work across a Transport Control Protocol/Internet Protocol (TCP/IP) network, with a router subsystem interconnecting the various subnetworks. This router subsystem may act as a firewall between the outside network and the internal subnetwork. The infrastructure of TIE-In works with these firewall routers by acting as a trusted agent through which access between users and applications can be accomplished by secure, authenticated, and controlled methods.

The Sandia network is comprised of primarily two subnetworks: the External Open Network (EON) and the Internal Restricted Network (IRN). The Master Front End System is located on the External Open Network. The EON has network connections to the Internet as well as to the Internal Restricted Network. Applications may be located on either the EON or the IRN. The Master Front End System also acts as the Kerberos Authentication Service for user and request authentication. The EON backbone utilizes an interconnection subsystem of routers to connect local area networks to the backbone. In some cases, such as with the interconnection of the IRN to the EON, these routers have been configured to restrict data connections. The restriction placed on these routers will only allow connections between certain systems or networks and limit the connection to specific network protocol ports. This feature provides additional security by limiting data connections to known paths. For some applications this capability may be a requirement.

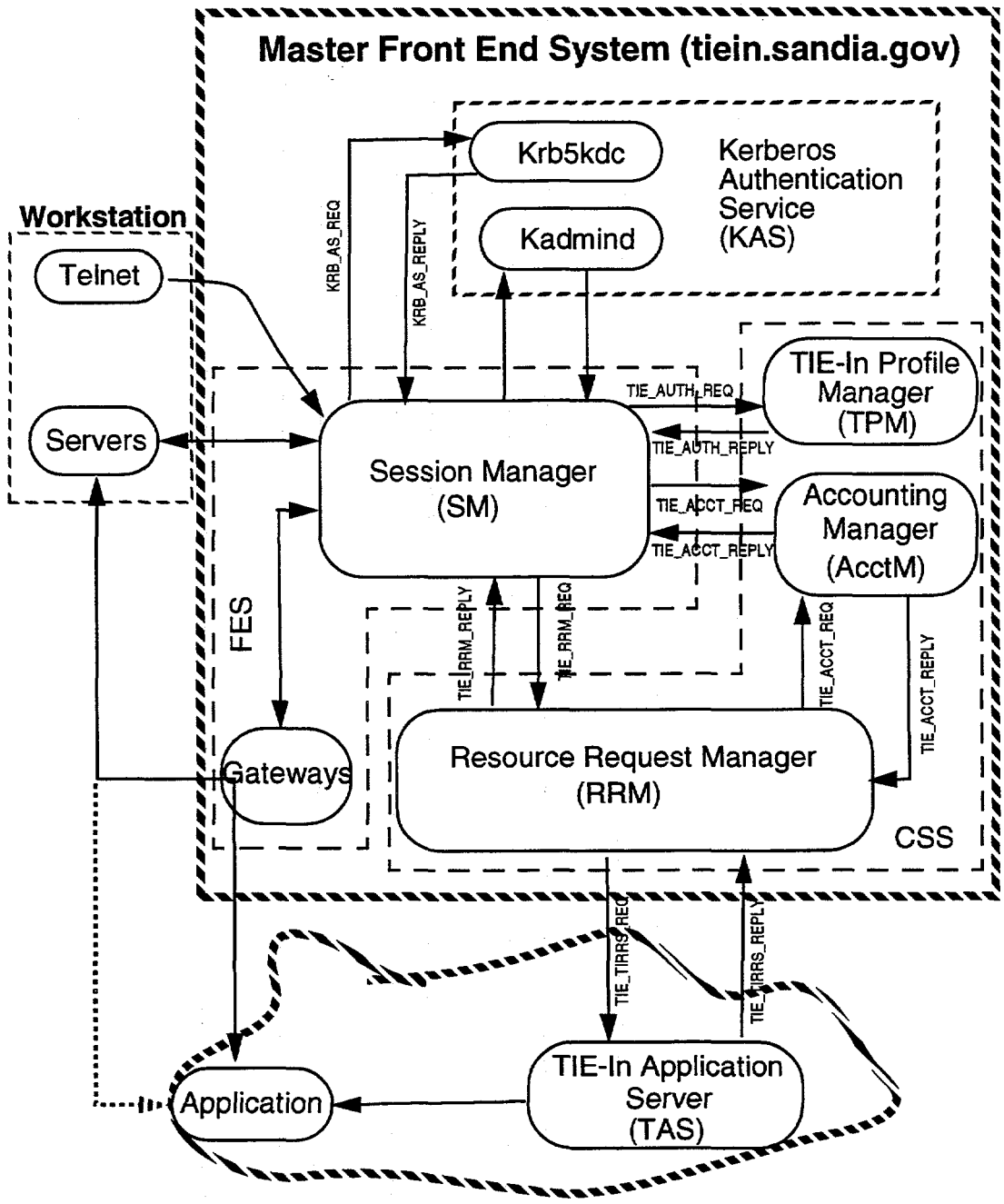


Figure 2. TIE-In system diagram.

The network protocols and ports used throughout the TIE-In System Infrastructure are defined and listed in Table 1. The Source/Process column reflects the source of the data and the process that created it. The Destination/Process column reflects the destination for the data and the process that will receive it. The last column reflects the actual port or ports to be used for the data transmission.

Table 1. System Infrastructure Network Protocols

Source/Process	Destination/Process	Network Protocol	Type	Port Value
WS/telnet	FES/SM	IP	TCP	23
FES/SM	WS/telnet	IP	TCP	>1023
FES/SM	WS/Xserver	IP	TCP	6000
WS/Xserver	FES/SM	IP	TCP	>1023
FES/SM	KAS/krb5kdc	IP	UDP	88
KAS/krb5kdc	FES/SM	IP	UDP	>1023
FES/SM	KAS/kadmind	IP	TCP	751
KAS/kadmind	FES/SM	IP	TCP	>1023
FES/SM	RRM	IP	TCP	1955
RRM	FES/SM	IP	TCP	>1023
MFES/RRM	FES/SMlog	IP	UDP	>1957
MFES/RRM	App/TAS	IP	UDP	1956
App/TAS	MFES/RRM	IP	UDP	1957
App/TAS	KAS/krb5kdc	IP	UDP	88
KAS/krb5kdc	App/TAS	IP	UDP	>1023
App/TAS	KAS/kadmind	IP	TCP	751
KAS/kadmind	App/TAS	IP	TCP	>1023
App/application	FES/X11GW	IP	TCP	6000-6100
FES/X11GW	WS/Xserver	IP	TCP	6000
WS/Server	FES/X11GW	IP	TCP	>1023
FES/X11GW	App/application	IP	TCP	>1023
FES/SM	MFES/TPM	IP	TCP	1956
MFES/TPM	FES/SM	IP	TCP	>1023
FES/SM	MFES/AcctM	IP	TCP	1957
MFES/AcctM	FES/SM	IP	TCP	>1023
MFES/RRM	MFES/AcctM	IP	TCP	1957
MFES/AcctM	MFES/RRM	IP	TCP	>1023

Table 1. System Infrastructure Network Protocols (Continued)

Source/Process	Destination/Process	Network Protocol	Type	Port Value
WS/mount	MFES/mnt_server	IP	TCP	< 1023
WS/mount	MFES/mnt_server	IP	UDP	< 1023
WS/nfs	MFES/nfs_server	IP	UDP	2049

where

WS	User's workstation/PC
FES	Front End System
MFES	Master Front End System
KAS	Kerberos Authentication Service
App	Application
SM	Session Manager
RRM	Resource Request Manager
TAS	TIE-In Application Server
TPM	TIE-In Profile Manager
AcctM	Accounting Manager
X11GW	X11 Gateway Service
TCP	Transport Control Protocol
UDP	User Datagram Protocol
SMlog	Session Manager Message Logging

Connecting to TIE-In

Access to TIE-In can be accomplished through the Internet or via a modem. For those users with direct access to the Internet, connecting to TIE-In is a two step process. First, the user will need to allow access to their local Xserver by the Front End System. This is accomplished on UNIX workstations by issuing the command

```
xhost tiein.sandia.gov.
```

On a PC, the X-Windows package normally provides some mechanism for maintaining an access control list. Simply add *tiein.sandia.gov* to this list.

The user can then use telnet to connect to *tiein.sandia.gov* (132.175.133.1). On a UNIX workstation, this is accomplished by issuing the command

```
telnet tiein.sandia.gov.
```

The method for initiating a telnet session on a PC depends upon the TCP/IP software being used, so users will have to consult their documentation. Prior to executing the telnet

command, UNIX users should ensure that their DISPLAY environment variable is set to their local workstation.

After the telnet connection is established, the Front End System will attempt to present an authentication window to the user. If it is unable to do so, it will prompt the user, through the telnet session, to enable access with the *xhost* command. If the authentication window still cannot be established, the user will be informed of the requirements for connecting to the Front End System and the telnet session will be closed.

Access from systems that do not have direct Internet connections can be accomplished through a Xyplex Terminal Server. The terminal server is enabled for Serial Line Internet Protocol (SLIP) and Point-to-Point Protocol (PPP). A bank of seven modems is connected to the terminal server. Users with the appropriate hardware and software can connect to the terminal server by dialing (505) 844-4414. The modems have a baud rate of 14.4K and can handle most industry standard protocols, including data compression. The terminal server requires a username and password, which is the same username and password required for accessing the TIE-In FES. Once connected, the terminal server will provide a prompt such as "132.175.133.2>" which indicates the IP address of the terminal server port the user is connected to. The user enables SLIP or PPP within the terminal server by issuing the command

`set port internet slip enable` or **`set port internet ppp enable`**.

At this point the connection acts as a node on the External Open Network. Some additional parameters that the user's software may need to know are shown in Table 2.

Table 2. TIE-In FES System Parameters

Parameter	Value
X11 Gateway's IP address	132.175.133.254
TIE-In domain name	.sandia.gov
IP addresses for the sandia.gov nameserver	134.175.109.2, 134.175.109.4, 132.175.133.1

To establish a connection to the Master Front End System, simply use the telnet command described above (**`telnet tiein.sandia.gov`**).

Charging Customers

TIE-In offers the applications the ability to charge its customers. The cost of running the application is set by the application administrator using the Session Manager. The

application can specify up to seven unique accounting types. Typical items for which to charge include CPU utilization, memory utilization, storage, and connection time. The application can also charge a monthly fee. The cost is defined on a per-unit basis. The number of units of each type is returned by the TAS process serving the application. The cost-per-unit is then multiplied by the number of units to obtain the cost for this accounting type. The costs for all defined accounting types are then added to obtain the total cost for the task.

The actual cost transfer from the application to the customer is accomplished using Sandia's Service Center Information System (SCIS). SCIS records are generated by TIE-In on a weekly basis. The records are then uploaded and run through the SCIS system. The charge number provided by the user when he ran the task is the one to which the cost will be transferred. If the customer is a Sandia employee, the charge number will simply be a CASE number. If the customer is not a Sandia employee, he must be with an organization that has either signed a User Facility agreement with Sandia or established a Work for Others (WFO) agreement with DOE and Sandia. The agreements authorize the user to run one or more specific TIE-In applications. Associated with the User Facility or WFO agreement is a Service Action Number that identifies the customer. All applications are assigned a Service Center Activity, which uniquely identifies the application to SCIS. The charge number for the user is then comprised of the concatenation of the Service Center Activity (four characters) and the Service Action Number (six characters). When a record with a CASE number of this type is processed by SCIS, it correctly transfers the cost of the task from the application to the customer.

TIE-In Security

This section discusses the security aspects of TIE-In. Security was designed into TIE-In from the beginning and has been strengthened throughout the development and maintenance phases of the project.

Authentication

Authentication verifies that a user is who he or she claims to be. It can be accomplished by asking for something the user knows (such as a username and password), something the user has (such as a SecurID card), or something biometrically unique to the user (such as a fingerprint or a retinal scan). Authentication is an essential part of the security of TIE-In and is performed by the Kerberos Authentication Service (KAS). The KAS authenticates a user by asking for a username and password and, in some cases, a SecurID passcode.

User Authentication

When a user attempts to log on to the TIE-In Front End System and establish a session, the user will be prompted for his or her username, password, and possibly a SecurID passcode. This information is used to authenticate the user. The user can be authenticated at one of three levels:

(1) Kerberos Version 5

(2) Kerberos Version 5 or SecurID

(3) Kerberos Version 5 and SecurID

The user's authentication level is set by a TIE-In administrator. If the user's authentication level is Kerberos Version 5 only, the user will be prompted for a username and password only. For the other two levels, the user will also be prompted for a SecurID passcode. If the user's authentication level is set to the second level, the user will be prompted to enter a SecurID passcode. The user may, however, choose not to enter a passcode. If the user's password is valid, he will be authenticated at the Kerberos Version 5 level only and a session will be established. In this case, the user will not be able to access any applications that require SecurID authentication, but will be able to access applications that only require Kerberos authentication.

The third level of authentication is the most secure. In this case, the user must enter a valid password and a valid passcode, or the authentication will fail. Thus, in order for an imposter to log in as this user, the imposter would have to know not only the user's password, but also the user's SecurID pin number. In addition, the imposter would have to physically have the user's SecurID card. All TIE-In administrators are required to be authenticated at this

level. This is simply because TIE-In administrators can view and change all information stored in the TIE-In database, so unauthorized access must be prevented.

Figure 3 shows a simplistic view of how a user who provided a SecurID passcode is authenticated. In this figure the user is the client. In part (a) of the figure, the user has provided his username, password, and passcode to the Session Manager. The Session Manager converts the user's password into a key, K_c. The user's SecurID passcode is then encrypted using K_c. The client's name (the username), the encrypted passcode, and a request for a Ticket-Granting Ticket (TGT) is sent to the Kerberos Authentication Service (KAS). The KAS looks up the client's key from its database and uses it to decrypt the passcode. The passcode is then sent to the SecurID server (SIDS) for verification.

Figure 3, part (b) shows the return path of the authentication procedure. If the passcode sent to the SIDS is valid for the user, the SIDS sends a reply to the KAS indicating that the passcode is valid. The KAS then creates a session key for use between the client and the KAS, $K_{C,KAS}$, and generates the requested Ticket-Granting Ticket, T_{tgt}. Within the T_{tgt}, the KAS sets a flag to indicate that the client has been authenticated with a SecurID passcode. It also stores $K_{C,KAS}$ in the Ticket-Granting Ticket. The T_{tgt} is then encrypted using the private key of the KAS, K_{KAS}. Both the session key and the ticket are then encrypted using the client's private key, K_c. The Session Manager can now decrypt the information returned by the KAS using the client's private key, K_c. The user has now been authenticated.

Figure 4 shows how the Session Manager can obtain network services on behalf of the user by using the user's T_{tgt}. An example of this is when the user requests to run an application. The SM must send a message, containing an appropriate ticket, to the Resource Request Manager.

In part (a) of Figure 4, the Session Manager is requesting a ticket to be used to request a service from the server. The Session Manager sends the T_{tgt}, encrypted in K_{KAS}, along with the server name and an authenticator, A. The authenticator contains information, such as a timestamp and the name of the user, which can be used to check for replay attempts and to validate the identity of the requestor. If the identity of the user in the authenticator does not match the identity in the ticket, the ticket request is denied. The request will also be denied if the timestamp in the authenticator is more than 5 minutes away from the current time (timestamp - current_time > 5 minutes). The authenticator is encrypted in the session key between the client and the KAS, $K_{C,KAS}$, while the T_{tgt} is encrypted in the private key of the KAS, K_{KAS}. When the KAS receives a request for a service ticket, it decrypts the T_{tgt} and the authenticator, verifies that the information matches, and checks the expiration time in the T_{tgt} to ensure that the ticket is still valid.

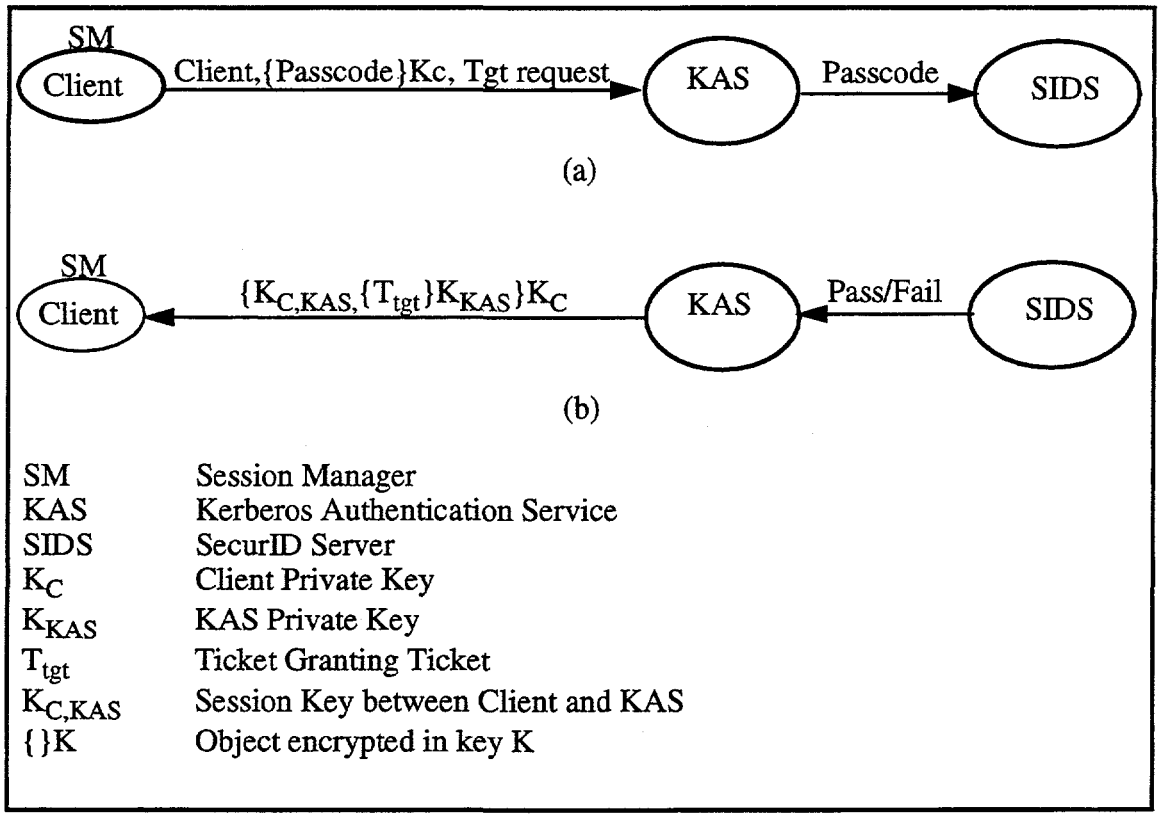


Figure 3. User authentication with SecurID.

After the KAS has verified the T_{tgt} and the authenticator, it generates a ticket for the server, T_{server}. As shown in Figure 4, part (b), T_{server} is encrypted in the server's private key, K_S. The KAS also generates the session key for the client and server, $K_{C,S}$, and places a copy of this key into T_{server}. Both T_{server} and $K_{C,S}$ are then encrypted in $K_{C,KAS}$, the session key between the client and the KAS. The client now has the ticket required to talk to the server.

Figure 4, part (c) shows the client requesting a service from the server using the ticket, T_{server}, obtained in the previous step. The client again generates an authenticator, encrypts it in $K_{C,S}$, and sends it, along with T_{server}, to the server. The server decrypts the ticket and the authenticator. It then verifies that the information matches, that the time limits have not been exceeded, and that the ticket is still valid. It then performs the requested service.

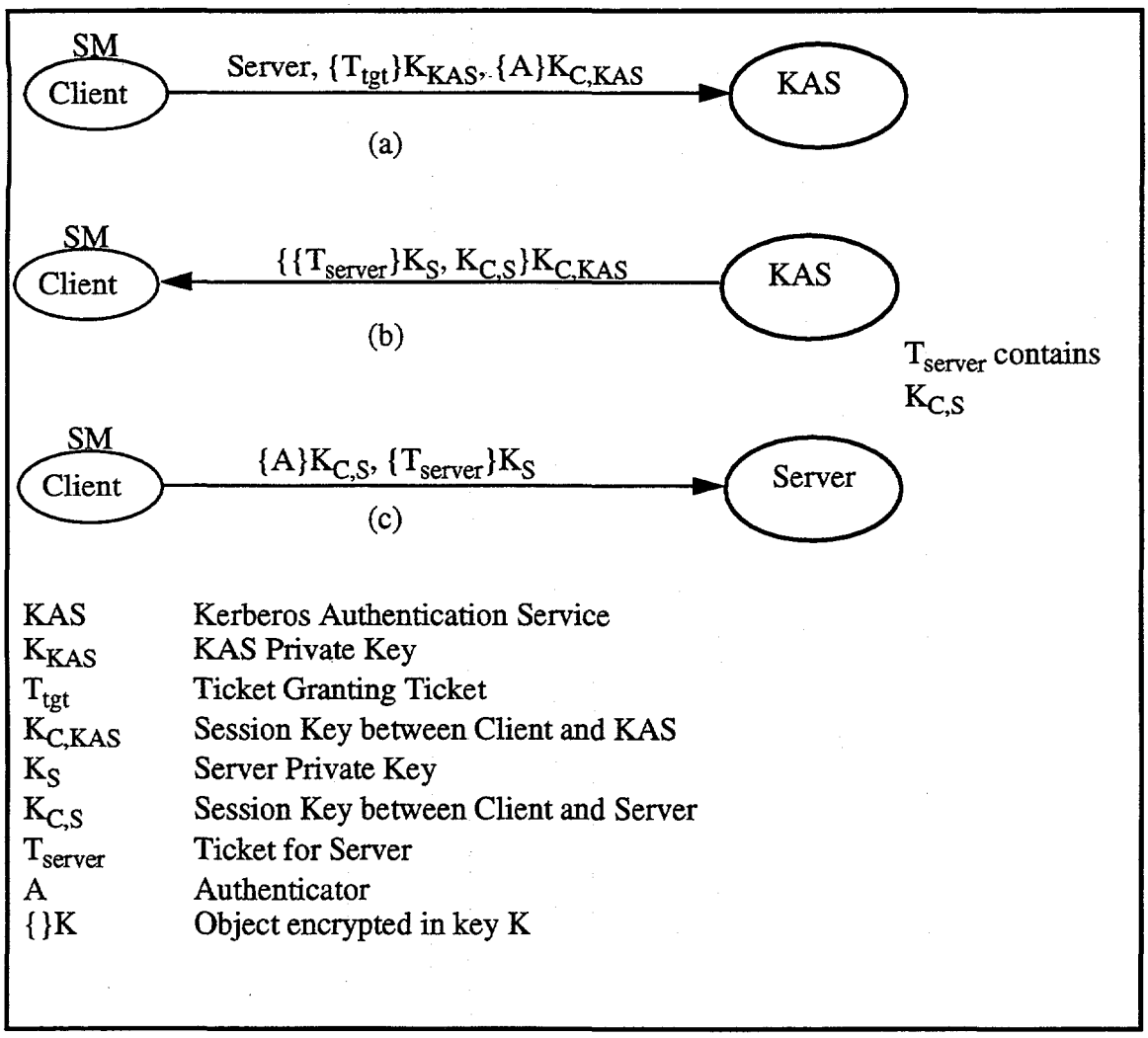


Figure 4. Obtaining services on behalf of the user.

Access Control

Access to an application can be controlled with a number of different methods. Each method is independent of the others. The main access control methods are

(1) requiring users to register for the application

(2) setting the user authentication level

(3) providing an access control list via the Front End System

(4) providing an access control list on the application server

Each of this methods will now be discussed in more detail.

User Registration

Application administrators have the option of requiring users to register for the application before being able to run it. When a user registers for an application, his request is sent to the TAS process on the application server. The TAS process will execute a registration script if one has been specified in the configuration file. In this script, the administrator can do a number of things to control access. For example, he can maintain a list of users who are allowed to register for the application. Users who are not on the list will have their request denied and will not be able to access the application.

If there is not a registration script specified in the configuration file but the application is set up to require users to register, a mail message will be sent to the application administrator when a user submits a registration. This allows the administrator to decide if the person should have access to the application on a user-by-user basis. For more information about the registration script, see "Description of an Add/Delete Account Request Message" in the "TIE-In Application Server Process Description" section.

User Authentication Level

The application administrator must specify the level at which users of the application must be authenticated. If the user is not authenticated at this level, he will not be allowed to access the application. The authentication level is set in two places: the application's entry in the Front End System database and in the configuration file on the application server.

The authentication level can be set to one of three levels, with the first level being the weakest authentication and the third level being the strongest:

(1) Kerberos Version 5
(2) Kerberos Version 5 or SecurID
(3) SecurID

If the user has been authenticated at a level greater than that required by the application, access to the application will be granted.

Both the application administrator and the TIE-In administrators can set the authentication level for the application on the Front End System. This authentication level is checked against the user's authentication level before any requests are sent to any of the application servers. If the user's authentication level is less than the required level, the user will receive an error message indicating that his level of authentication was too weak.

Even if the user has been authenticated at the proper level as specified in the Front End System, the user must also have been authenticated at the appropriate level as specified for the application in the configuration file. Since this file resides on the application server, only the application administrator can modify the required authentication level. If the application can run on more than one server, each server has a configuration file and can therefore set the required authentication level for the application on that server. This allows one server to require only Kerberos authentication, while another server can require SecurID authentication. If the user tries to run the application but has not been

authenticated at the level specified in the configuration file, he will receive an error message indicating that he does not have permission to run the application.

Access Control List via the Front End System

The TIE-In Front End System offers the application administrator the option of creating an access control list for the application. If a user is not on the access control list, he will not even be aware that the application exists. He will not be able to get information about it, register for it, or attempt to run it. When the administrator wants to give access to a user, he can simply add him to the access list. The user can then register for the application, and if the registration is successful, he can then run the application.

Access Control List on the Application Server

The access control list on the Front End System is a good method for limiting access to an application. The one disadvantage of this method is that TIE-In administrators can also add users to the access control list of an application. While this does not really pose a problem, a method exists that gives the application administrator final authority over which users can run the application. This method is to provide a local authorization script, which will be executed by the TAS when a user requests to run the application.

The application administrator can use the local authorization script to maintain an access control list on the server, independent of the access control list on the Front End System. Thus, even if a user is on the access control list on the Front End System, he must also be on the local access control list or his request will be denied.

An application can use the local authorization script in conjunction with the registration script to maintain an access control list. When a user registers for an application, his name can be added to the access control list. When the user then runs the application, his name can be verified against the access control list before allowing him to run the application. When the user removes the application from his list, his name can then be removed from the access control list. This is a simple method to ensure that anyone attempting to run the application has registered for the application, without being too restrictive about who can access the application.

For more information about the local authorization script, see "The Local Authorization Policy" in the "TIE-In Application Server Process Description" section.

The X11 Gateway

The TIE-In Front End System and all of the associated TIE-In applications use X Windows for their Graphical User Interfaces. This provides a common presentation method between all of the applications. A couple issues had to be resolved, however, before TIE-In could use X in a seamless manner.

In the current Sandia computing environment, systems that reside on the External Open Network (EON) or on a network external to Sandia cannot access a user's X server if it resides on the Internal Restricted Network. The X packets are blocked at the packet-filtering router, commonly known as the diode, separating the EON from the IRN. Therefore, a mechanism had to be developed to allow seamless flow of X Window data between the EON and the IRN.

The second issue related to using X Windows is that most X11 servers require the user to specify all systems that are allowed to access the user's display. Alternatively, the user can specify that all X11 clients can access his display. The latter case is highly undesirable, because it allows any system to access the user's display. This can be a problem because it is relatively easy to capture a user's display once given access to it. Thus, an intruder can easily capture the user's display without the user being aware of the attack. This potential security hole can be closed by having the user specify only those systems that are allowed to access his display. This is typically done with the *xhost* command on UNIX systems, and with an access list on most PC X-Windows packages. This approach, however, would require the user to know all of the systems on which the various TIE-In applications reside.

Both of the above mentioned problems are solved by using an X11 gateway. When a user selects an application through the Session Manager, an X11 gateway is established. All applications send X11 data to the gateway, which then forwards it onto the user's X11 server. The diode separating the EON from the IRN has been configured to allow the TIE-In machine, on which the gateway processes run, to access X11 servers that reside on the IRN. Thus, seamless access between the EON and the IRN is achieved. In addition, using the gateway means that only the TIE-In machine needs to be given access to the user's X11 server.

Use of the X11 gateway does incur a performance penalty of 10 percent to 50 percent. Thus if a user does not want to use the gateway for performance reasons, the capability exists to easily disable it through the main TIE-In window. Efforts are underway to reduce this performance penalty.

Magic Cookies

When access is granted to an X11 client, all users on the client are given access. Thus, it would be possible for an unauthorized user on the client to access the user's X11 server. To solve this problem, TIE-In implemented the use of magic cookies. The concept of magic cookies was developed at MIT and is included in X11 release 5 (X11R5).

When a user runs an application, the X11 gateway process creates a magic cookie, which is simply a 32 character key. The magic cookie is then associated with the user and with the proxy display created for the user by the gateway. This information is sent from the X11 gateway process to the Session Manager. The Session Manager then forwards the information to the TIE-In Application Server (TAS) process running on the application server. The TAS appends the proxy display and the magic cookie to a file named */tmp/.Xauthority_username*, where the username is the TIE-In username of the user. If the file does not exist, it will be created. The permissions on the file are set so that only the user

has read/write permission. Thus another user on the system cannot view the file and obtain the magic cookie.

Before the TAS executes the application, it sets two environment variables: DISPLAY is set to the proxy display created for the user, and XAUTHORITY is set to the user's .Xauthority file. When the application is executed, it reads the magic cookie from the .Xauthority file for the proxy display and presents it to the gateway process for authentication. If it matches, the application is allowed to access the user's X11 server. If it does not match or if it is an empty cookie, access to the user's server is denied. In this manner, authentication of the X11 client is brought down to the user level.

Notification

TIE-In offers the user the ability to require notification of X11 access. When this option is used, a notification window will appear on the user's display prior to any connections to the user's X server by any of the TIE-In applications. The notification window asks if the user wishes to allow the access. By default this feature is turned off, since it can be confusing to users.

Interactive Collaborative Environment

Incorporated into the X11 gateway service is the Interactive Collaborative Environment (ICE). With ICE, users may specify X displays with which they wish to share the application. This allows for collaborative design and interactive assistance with experts of TIE-In applications.

Logging

Extensive logging is performed by TIE-In. The messages generated by logging are helpful in determining the activity on the system, watching for problems, and problem debugging. The types of logging performed by the various TIE-In processes are discussed below.

Syslog

All of the TIE-In processes utilize the syslog mechanism for performing extensive logging. The processes that perform logging include the Session Manager, the Resource Request Manager, the Profile Manager, the Accounting Manager, and the three processes that comprise the Common File System. The X11 gateway service and the Kerberos authentication service log messages as well.

All messages are written to a common file, */usr/adm/tiein.log*, on the Front End System. By using the UNIX *tail* command, TIE-In administrators are able to view the messages that are written to this file as soon as they are written. Thus, if there is a problem, it is usually quickly detected by a TIE-In administrator.

Various types of messages are logged by the different processes. The Session Manager logs a message when a user makes a connection to the Front End System, when a user

successfully logs in (thus initiating a session), and when a user logs out (ends the session). The RRM, TPM, and AcctM all log messages when they initiate or receive a request and when they send or receive a reply. Included in the log message is the type of request or the status of the reply. Thus it is very simple to determine what requests are being sent and if the request was successfully serviced. These processes also log a steady state message, which is simply a message to indicate that the process is still running. The X11 gateway service logs a sequence of messages when establishing a proxy display on behalf of the user. It also logs a message when the proxy display is closed.

The Kerberos Authentication Service logs a variety of messages. The most useful messages are those that indicate when the Key Distribution Center (KDC) has issued a service ticket and when a user has failed authentication. The most common cause of an authentication failure is when the user has forgotten or incorrectly typed his or her password. If multiple failures occur within a short time, an administrator will usually call the user. This is to ensure that the user has been trying to log in, and if so, has forgotten his password. If he has, a TIE-In administrator can easily reset it for him.

The messages written to the log file provide a very useful mechanism used by the TIE-In administrators to get a current picture of the state of the system. They also make it easy to quickly detect any errors, thereby increasing the reliability of the system.

TAS Logging

The TIE-In Application Server process also performs logging. If it is able to write to syslog, it does so. In this case the file resides in */usr/adm/tirrs.log* on the application server. If it cannot write to syslog, an alternate method is available to write to stdout. To accomplish this, the application administrator should simply kill the *tirrs* processes, set the TIE_TIRRS_DEBUG environment variable to 1 (one), then restart the *tirrs* process and redirect the output to a file.

The TAS logs a message when it receives a request, when it sends a reply, and when an error occurs. It also periodically logs a steady state message. This information can be very useful in debugging a problem.

For more information on the TAS, please see the "TIE-In Application Server Process Description" section.

Change Logs

Another important logging feature of TIE-In is the tracking of who made what changes to any user information, application information, or project information. These logs show what fields were changed, by whom, and when the changes were made. Whenever an update is performed, the Profile Manager writes a message to a file in the */usr/adm* directory on the Front End System. One file is for changes made to user information, one file for changes made to application information, and one file for changes made to project information. These logs can be helpful in auditing or in resolving any conflicts over who made what changes and when the changes were made.

The authority to make changes is tightly controlled by the TIE-In system. Normal TIE-In users can only make changes to a limited number of fields in their own user record. An application administrator can make changes to any applications for which he is listed as the administrator. Similarly, changes to projects can only be made by a user who is listed as the project manager for the project. A TIE-In administrator can make changes to any user, application, or project information.

Session Logging

When a user logs into the TIE-In Front End System, a session is initiated for that user. When a session is initiated, information about the session is written to a database table. The entries in the table can only be viewed by TIE-In administrators. Administrators use this information to determine who is logged into TIE-In. The information stored in the database table and displayed to the administrator is shown in Table 3.

Every session is assigned an identifier that uniquely identifies the session. The identifier can be used to retrace a user's actions during the session, including what applications he ran. The session identifier is composed of a number of fields that, when combined, will uniquely identify the session for all time. The format of the identifier is

user_display:session_pid:start_date:start_time

where *user_display* is the IP address of the user's X-Window display, *session_pid* is the process id of the session on the Front End System, *start_date* is the date the session was initiated, and *start_time* is the time the session was initiated.

Table 3. Current Session Information

Field Name	Description
session_id	the session id of the session
user_name	the username of the owner of the session
user_display	the IP address of the user's display
fes	the name of the Front End System to which the user logged in
start_time	the time the session was started, which is the time the user logged in

When a user exits the TIE-In system, the information stored in the database about the user's session, along with the time the user exited the session, is logged to a file. This file can then be used by an administrator to obtain information about all completed TIE-In sessions.

Security Scripts

A number of scripts are run automatically on the tiein machine (tiein.sandia.gov) that extract pertinent information from the log files and mail the information to the TIE-In administrators. The information extracted includes

(1) the names of systems outside of the Sandia environment that made connections to TIE-In,

(2) the names of all users who logged into the TIE-In Front End System during the current week,

(3) the names of all users who logged into the TIE-In terminal server during the current week,

(4) the IP addresses of systems who made a connection to the tiein machine with the source route option set, and

(5) detailed statistics about which users ran what applications at what time.

With the exception of the detailed statistics, the above information is generated and mailed to the TIE-In administrators once a week. The detailed statistics are mailed once a week. This information simply helps the administrators determine who has been using TIE-In, what applications they are running, and when they are running them.

Information Verification

Employee Verification

When a Sandia employee submits a TIE-In registration, the user's social security number is verified against a file containing all current Sandia employees. If a match is not found, the user's registration is rejected. The most recent version of the employee file is obtained daily to minimize any lag time for new employees. If the user is determined to be a Sandia employee, the registration is accepted.

Each night, all registered TIE-In users that Sandia employees or contractors are checked against the employee file to ensure that they are still an active employee/contractor. If a user is determined to no longer be an active employee or contractor, his account is disabled and a mail message is sent to all TIE-In administrators. Once the user's account is disabled, he will not be able to log in. After receiving the mail message, a TIE-In administrator will investigate and delete the user's account if necessary.

Nonemployee Verification

For those users who are not Sandia employees or contractors, the information provided on the registration form is verbally verified with the user. In addition, each of these users must have a Sandia sponsor who can verify their need to use the TIE-In system and provide reasonable assurance that the user is trustworthy. Periodically, these users and their sponsors are contacted to determine if they still require a TIE-In account. If not, the account is deleted.

All users who will be accessing applications on the Internal Restricted Network must have a SecurID card. External users requesting a SecurID card must complete a Central Computing Resource Request form, sign it, and return it to a TIE-In administrator. The TIE-In administrator will then obtain the signature of the supervisor of the user's Sandia sponsor.

Charge Account Verification

All users are required to provide a valid charge account before being able to access any applications. When a user inputs a CASE number, it is verified against the list of all valid CASE numbers. The most recent version of the valid CASE number file is obtained nightly from Sandia's finance organization to ensure that the latest information is being used. Each night, all CASE numbers stored in the TIE-In database are checked against the valid CASE file. If a CASE number is no longer valid, all users who have provided that CASE number will be required to input a new, valid CASE number. This process ensures that all CASE numbers being used are valid.

Projects

TIE-In uses the concept of projects to help organize external customers. The driving force behind the creation of projects was the method that TIE-In uses to charge external customers. External customers must sign a contract with Sandia and provide money at the time of entering into the contract. At this time, a project identifier is created for the users from the external organization. Associated with the project identifier is the dollar amount of the contract, the charge account to be charged, and an access control list. The access control list specifies which users are allowed to charge to the project identifier. The charge account number is typically a service action number created in the Sandia business office at the time the contract was signed. This number will be used to transfer costs from the application to the customer. The dollar amount of the contract is decremented each time a project member runs an application that charges its users. The dollar amount is also checked before a project member can run an application to help ensure that there is enough money left to cover the cost of running the application.

All external customers must have a project identifier in order to run any of the TIE-In applications. This helps to ensure that all external customers are representing a valid external organization and are truly interested in one or more of the applications available through TIE-In.

Session Manager Process Description

The Session Manager (SM) is the interface between the user and the rest of the components of TIE-In. The SM communicates with all other manager processes, as well as with the Kerberos Key Distribution Center and the X11 Gateway. The message formats for the exchange of information between the SM and the other components of TIE-In are provided in Appendix A: Message Formats. The following sections describe the various functions performed by the SM.

User Authentication

When the user first connects to TIE-In, a session is created for the user. This session is controlled by the Session Manager process. Prior to establishing the session, the Session Manager uses the access control file */tiein/etc/gw_acl* to verify that the display with which it is about to establish a session is allowed. The access control file contains a list of systems and networks that are allowed to access TIE-In.

The first thing the user must do in the session is present his username and password. The SM takes this information and submits a request to the TIE-In Profile Manager (TPM) for information about the user. If the TPM does not return any information about the user, the user is denied access to TIE-In. Otherwise, the SM sends the information returned from the TPM to the Kerberos Authentication Service (KAS). The KAS attempts to validate the information and, if successful, returns a Kerberos ticket to be used in subsequent application requests by the user. From the profile information returned by the TPM, the SM determines if the user has a hardware authentication device (SecurID card). If so, the user is prompted to enter the passcode from that card. If he does not provide the passcode, he may still be logged in but will be denied access to any applications that require hardware authentication of its users. If the user enters the correct passcode, he is logged in and presented with the main TIE-In window.

The Main Window

The main TIE-In window is shown in Figure 5. This is a selection-based window listing all the facilities that the user may access. The window has three main components:

- The upper component is the authorized application list. This list contains the applications available to the user.
- The middle component allows the user to perform general administration functions such as changing his Kerberos password, obtaining status and accounting information on current and prior tasks, modifying his profile information, and turning on or off the X11 Gateway or X11 access notification.

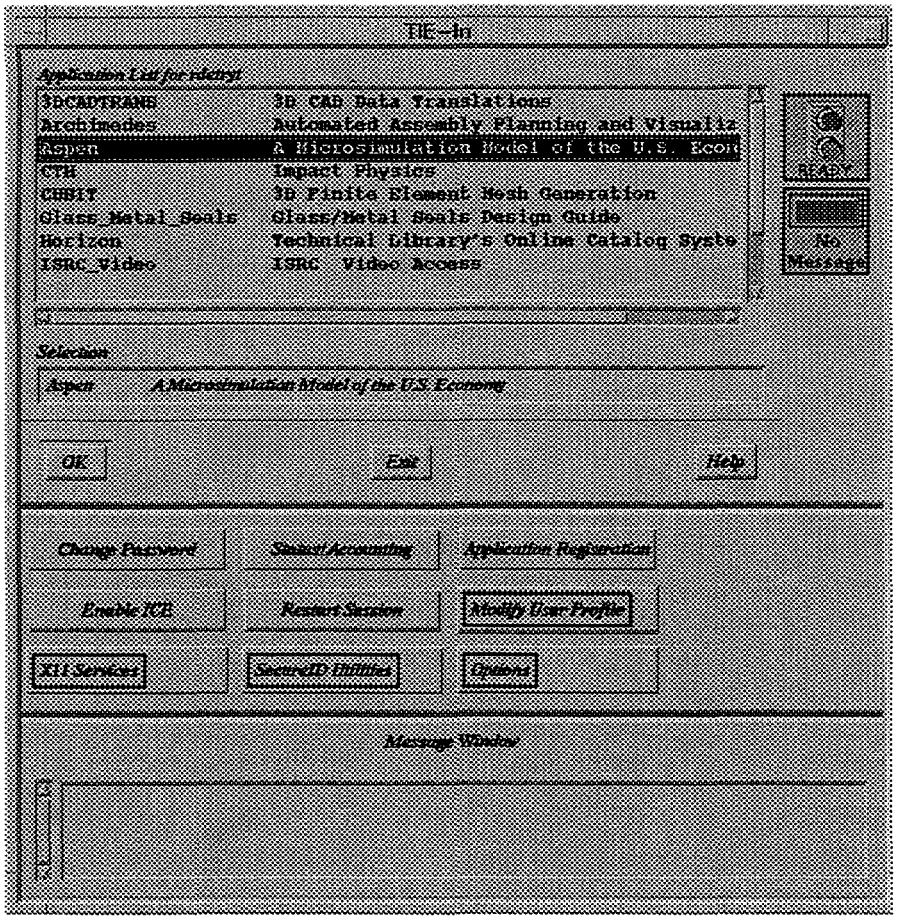


Figure 5. The main TIE-In Window.

- The third component of the main window, located on the bottom, is the message area for error and informational messages. This area is equipped with scrollbars, allowing the user to scroll back and examine earlier messages.

Running an Application

The most common use of the Session Manager is to submit a request to run an application. When a user selects an application to run, the SM formats a request and sends it to the Resource Request Manager (see the "Resource Request Manager Process Description" section for more details). The RRM responds back with a status message. If the request was successfully processed, the application will soon appear on the user's window and a message will be written to the message area on the user's TIE-In screen. If the request was not successfully processed, an error message will be displayed in a separate information window on the user's screen. Once the application appears on the user's screen, the TIE-In

window is no longer necessary, but it can be used to obtain status information and perform administrative functions, as discussed below.

Obtaining Status and Accounting Information

From the main window, the user can obtain status and accounting information about active and completed tasks by selecting the Status/Accounting button (see Figure 5). When the user presses this button, the SM issues a message to the Accounting Manager asking for the information requested by the user. The user can request information on active tasks, completed tasks, archived tasks, or all tasks. He can limit the search by selecting a time period, selecting a particular application, or requesting information about a specific task. The Accounting Manager returns the information to the SM, which then creates a status window on the user's display.

Modifying the User Profile Information

The TIE-In Front End System maintains information specific to each user, such as full name, address, phone, e-mail address, etc. This information is referred to as the "user profile." Through the Session Manager a user can modify some of his profile information, such as phone number, e-mail address, etc. When the user is done modifying his profile, the SM sends the updated information to the TIE-In Profile Manager.

Adding or Removing Applications

When a user logs in for the first time, he will be presented with a list of applications that are open for use by all registered TIE-In users. Applications can be added or removed from this list by selecting the Application Registration button on the main TIE-In window. A new window is displayed on the user's screen that allows applications to be added or removed. When the user selects an application to be added, the Session Manager sends a request to the Resource Request Manager. The RRM in turn sends a request to the appropriate TIE-In Application Server, which will execute an add account script for the application if one exists. This script may or may not add the user. Many applications are not open to all TIE-In users, and therefore may deny a user an account. If this occurs, a message indicating that the user's account could not be added is propagated back to the SM and displayed on the user's screen. In addition, a mail message will be sent to both the application administrator and the TIE-In administrator indicating that the user failed the accounting script. If the user's account was successfully created, the application is added to his application list.

When a user requests the removal of an application from his application list, the Session Manager again sends a request to the Resource Request Manager. The RRM in turn sends a request to the appropriate TIE-In Application Server, which will execute a delete script

for the application if one exists. A success or failure message is then propagated back to the user. If successful, the application is removed from the user's list.

User Registration

The Session Manager provides the interface for a user to submit a registration for a TIE-In account. The user simply fills in an electronic form that asks for the user's full name, address, phone number, etc., and the Session Manager sends the information to the Profile Manager for storage. The Session Manager then notifies the TIE-In administrator, via a pop-up window, that a registration has been submitted.

The TIE-In administrator uses the Session Manager to register a user. The Session Manager displays a form that contains the user's information and prompts the administrator to enter a username and userid for the user. The administrator can also enter additional information, such as an account expiration date, the user's realm, the user's home directory, etc. When the administrator is done entering the information, the Session Manager sends a request to the Profile Manager to register the user. The Profile Manager then places the information in the TIE-In database.

Resource Request Manager Process Description

The Resource Request Manager (RRM) provides the network interface between the applications and the Front End System. The RRM communicates with the Session Manager (SM), the Accounting Manager (AcctM), and the TIE-In Application Server (TAS). The messages sent between the processes are described in Appendix A: Message Formats. The types of requests sent from the SM to the RRM, from the RRM to the TAS, and from the RRM to the ACCTM are described in Tables 4, 5, and 5, respectively.

Table 4. Request types sent by the SM to the RRM

Request Type	Request Description
RRM_SUBMIT_REQ	sent when the user initiates a request to run an application
RRM_STATUS_REQ	sent when the user inquires about the status of an active task
RRM_DELETE_REQ	sent when the user wants to delete an active task
RRM_ADD_ACCT	sent when the user adds an application to her application list
RRM_DEL_ACCT	sent when the user removes an application from her application list

Request Processing

When the RRM receives a request from the Session Manager (Table 4), it sends a similar request to the appropriate TAS (Table 5). The request is then placed on a waiting list until a reply message is received from the TAS. The TAS replies to the RRM and indicates the request was processed successfully or that an error occurred. In either case, the status of the request is sent back to the SM and on to the user. The request is then removed from the waiting list or placed on an active task list that is maintained by the RRM.

Table 5. Request types sent by the RRM to the TAS

Request Type	Request Description
TIRRS_SUBMIT_REQ	sent when the RRM receives a submit request from the SM
TIRRS_STATUS_REQ	sent when the RRM receives a status request from the SM
TIRRS_DELETE_REQ	sent when the RRM receives a delete request from the SM
TIRRS_ADD_ACCT	sent when the RRM receives an add account request from the SM
TIRRS_DEL_ACCT	sent when the RRM receives a delete account request from the SM
TIRRS_COMPL_ACK	sent when the RRM receives a message from a TAS indicating that a task has completed. This request is used to inform the TAS that the RRM received the final information.

Submit Request

When the user selects an application from his application list, the SM sends a submit request (RRM_SUBMIT_REQ) to the RRM. If the request is processed successfully by the TAS, the RRM notifies the Accounting Manager that a new task has been submitted by sending it a ACCT_ADD_REQ message. The RRM will periodically receive updates from the TAS containing the status and the latest accounting information for the task. This information is also sent to the Accounting Manager through a ACCT_UPDATE_REQ message. When the task completes, the TAS sends a message to the RRM that contains the final accounting information. The RRM removes the task from its active task list and sends the accounting information to the Accounting Manager.

Delete Task Request

Through the Session Manager, the user is able to delete an active task. This is typically unnecessary since the user is usually able to exit from the application. The situation may arise, however, that the application and the user's X server break communication and the application does not immediately recognize that the connection has been lost. In this case the user may use the Session Manager to delete the task. When this option is selected, the SM sends a delete task request (RRM_DELETE_REQ) to the RRM. The RRM passes the request to the appropriate TAS by sending it a TIRRS_DELETE_REQ request. The TAS sends a hang-up signal to the specified task and returns a status message indicating that the delete was successful or that an error occurred. This information is then sent back to the

SM and on to the user. If the delete is successful, the RRM will soon receive a status message from the TAS containing the final accounting information for the task. This information is then sent to the Accounting Manager.

Add Account Request

When the user attempts to add an application to his application list, the SM sends an add account request (RRM_ADD_ACCT) to the RRM. The request is reformatted and sent to the appropriate TAS in a TIRRS_ADD_ACCT request. If the application has an add account script, it is executed by the TAS and the status (success or failure) is returned to the RRM, which passes it back to the SM. If the user's account was successfully added, the application will be placed on the user's application list. Otherwise, a message is returned to the SM indicating that the user's account could not be added.

Delete Account Request

When a user removes an application from his application list, the SM sends a delete account request (RRM_DEL_ACCT) to the RRM, which reformats it and sends it to the appropriate TAS via a TIRRS_DEL_ACCT request. If the application has a delete account script, it is executed by the TAS and the status (success or failure) is returned to the RRM. This status is then passed back to the SM and the user. If the user's account was successfully deleted, the application is removed from his application list.

Table 6. Request types sent by the RRM to the AcctM

Request Type	Request Description
ACCT_ADD_REQ	sent when a new task has been successfully submitted to an application
ACCT_INFO_REQ	sent when the RRM requires information about the current state of a task
ACCT_UPDATE_REQ	sent when the RRM receives an update message from the TAS

Handling Abnormal Cases

After submitting a request to the TAS, the RRM waits for a reply. If it does not receive a reply from the TAS in a specified time, it reissues the request. If no response is received after three attempts, the SM is notified that the resource is not available. The RRM may initiate its own status request if it has not received, within the last 30 minutes, a status message from the TAS. This will handle the abnormal case when the TAS returns a status or reply message that is not received by the RRM.

If the RRM receives an update from a TAS for a task that is not listed on either the active or the waiting lists, the RRM will ask the Accounting Manager for the current state of this task by submitting a ACCT_INFO_REQ request. If the Accounting Manager has no record of this task, then the message from the TAS is ignored. If the Accounting Manager has a record for this task, the new information will be passed back to the Accounting Manager. If the message from the TAS is an update, the task is added to the active task list. The RRM will maintain file copies of the active list and waiting list for recovery purposes.

In the event that the Accounting Manager does not respond to a message sent by the RRM, the RRM will place the message on a queue. When the Accounting Manager is restarted, the RRM will send all of the messages on the queue in the order that they where originally received. This prevents any status or accounting information from being lost.

Gateway Service Process Description

The Gateway Service is a server process that establishes gateways for a variety of network communication protocols. The program, based on Digital Equipment Corporation's public domain *xforward* code, allows for the creation of proxy gateways for X, Transport Control Protocol (TCP) and User Datagram Protocol (UDP), and the Network File System (NFS) (through the use of the Common File System). The use of a gateway allows network communication in a more secure environment. For Sandia, the gateway allows network communication to take place between the Internal Restricted Network and the External Open Network. The Gateway Service is based on the typical client/server model, in which clients issue requests to the Gateway Service, which performs the request and returns the results of the operation to the client. In the TIE-In infrastructure the clients are the Session Manager and Request Manager. However, the Gateway Service can also be used by clients outside of TIE-In. At Sandia, the program *xallow* can be used to make requests to the Gateway Service to create an X proxy, so that X window applications on the EON can communicate with displays on the IRN. The protocol used to communicate with the Gateway Service is provided in Appendix A: Message Formats.

Gateway Requests

A request made to the Gateway Service consists of the following information:

(1) Authentication Credentials
(2) Server
(3) Operation
(4) List of Clients

Authentication Credentials:

These are tokens that identify and authenticate the user making the request. The protocol provides a type field as part of the credentials, so that any number of authentication types could be used. However, the only supported type is Kerberos Version 5.

Server:

For all gateway types other than CFS, this consists of the string "server:port" where "server" is either the system name or IP address of the server, and "port" is the TCP or UDP port the server is listening to. This identifies the network connection to which UDP packets or TCP connections sent to the proxy will be forwarded. By default, the server and the client sending the gateway request must be the same unless special exceptions are given to the client through the GW_systems_acct entry in the */tiein/etc/gw_acl* file. Session managers

and Request Managers are given special exception since they must create and remove X proxies between various applications and displays.

For a gateway type of CFS, the server filed will contain the string "NFS_directory:application:username:userid." The NFS_directory is the user's home directory to be mounted by the specified application. The username and userid are the user's TIE-In user name and userid.

Operation:

The operation actually specifies the type of gateway(proxy) to establish, and some of its characteristics. The following is an annotated list of the available operations:

- Create an X11 proxy with a notification window.
- Create an X11 proxy without a notification window.
- Create an X11 proxy with a notification window and using a generated MIT magic cookie for user authorization. See "Magic Cookies" in the "TIE-In Security" section.
- Create an X11 proxy without a notification window but using a generated MIT magic cookie for user authorization.
- Create a shared X proxy (ICE).
- Create a UDP proxy.
- Create a TCP proxy.
- Create an NFS proxy - For this proxy the server consists of the string "NFS_directory:application:username:userid" For more information on the NFS gateways see the "TIE-In Common File System" section.
- Close down any proxy gateways associated with the passed server and the user identified in the request. Closing a proxy means that once all connections to the proxy are terminated, the proxy is destroyed.
- Terminate any proxy gateways associated with the passed server and the user identified in the request.

The Gateway Service can restrict which systems and/or networks can create a particular type of gateway. This is done by way of the */tiein/etc/gw_acl* file. This file contains entries for each operation, and restrictions can be placed on networks or particular systems in a network as to whether that site can create a particular proxy gateway. For example, the following entries indicate that all of network 134.253.14 can create an X proxy gateway except 134.253.14.205.

GW X11 Proxy Create:134.218.14.0:
-GW X11 Proxy Create:134.218.14.205:

List of Clients:

This is the list of hosts or IP addresses that will be allowed to connect or send data to the proxy gateway. All other systems will be denied access to the proxy.

Gateway Reply

After processing the request the Gateway Service sends back a reply (see "Gateway Service Reply" in Appendix A: Message Formats). The reply will indicate whether or not the request was successful and will contain a reply string which, in most cases, will contain the "IP_address:port" of the proxy server. For secure X proxies using magic cookies, the string will be in a format suitable for the *xauth* utility. For NFS proxies the string will consist of the proxy NFS mount point to be used by the application.

Intentionally Left Blank

Accounting Manager Process Description

The Accounting Manager (AcctM) maintains the database of active, completed, and archived tasks. The manager listens on TCP port 1957 for TIE_ACCT_REQ messages from the Session Manager (SM) or the Resource Request Manager (RRM). The message formats for the TIE_ACCT_REQ and TIE_ACCT_REPLY messages are provided in Appendix A. The Accounting Manager will only respond to messages originating from a valid Front End System or an authorized RRM process. Three types of messages may be submitted to the AcctM: add (ACCT_ADD_REQ), info (ACCT_INFO_REQ), and update (ACCT_UPDATE_REQ).

Add Requests

An add request is submitted to the AcctM by the RRM whenever a user successfully submits a task to a TAS. Information about the task is stored in the database. For an add request, credentials must be provided with the request to authenticate the originator of the request.

Information Requests

Information requests are submitted by the SM on behalf of a user or administrator. Users can extract information based on the application, task state (active, completed, or archived), start and stop times, or a specific task id. Additional search fields are provided for application and TIE-In system administrators. Application administrators are restricted to retrieving information about tasks that have been submitted to the application under their administration. TIE-In administrators can retrieve information about any task. Authorization does take place to ensure that a user can only view information about his own tasks and an application administrator can only view information about his own application.

Update Requests

Update requests are submitted to the AcctM by the RRM. The RRM periodically receives information about tasks from the various TIE-In Application Servers and passes the information on to the AcctM for storage in the database. If the task has completed, the AcctM moves it from the active task list to the completed task list.

Database Tables and Information

The specific information stored in the database is shown in Appendix D: The TIE-In Database Tables. The tables involved are *active_tasks*, *active_units*, *completed_tasks,* and *completed_units*.

TIE-In Profile Manager Process Description

The TIE-In Profile Manager (TPM) maintains the database describing the characteristics of TIE-In users and resources. The manager listens on TCP port 1956 for TIE_AUTH_REQ messages from the Session Manager (SM) or the Resource Request Manager (RRM). The message formats for the TIE_AUTH_REQ and TIE_AUTH_REPLY messages are provided in Appendix A: Message Formats. TIE_AUTH_REQ messages can be submitted to the TPM with any of the following request types: add (AUTH_REQ_ADD), info (AUTH_REQ_INFO), or delete (AUTH_REQ_DEL). For add or delete requests, credentials must be provided in order to authenticate the originator of the request.

TPM Add Request

The Session Manager submits AUTH_REQ_ADD request-type messages to the TPM whenever information about a user, an application, or a project is created or modified. In addition, whenever a user logs into the Front End System (initiates a session), information about the session is sent to the database for storage. Note that an add request is used for updates as well as for adding new information. When an add request is received, the TPM determines if there is already an entry in the database for the specified user or application, and, if so, performs an update rather than an add.

The TPM determines which type of information is to be added or updated by examining a field in the TIE_AUTH_REQ message called *nametype*. Table 7 describes the nametypes used in association with an add request and for what purposes they are sent. In the actual request, the nametype is prefixed by TIE_AUTH_NAMETYPE_.

Table 7. Description of TPM add request nametypes

Nametype	Description
USER	sent when a TIE-In administrator registers a user and whenever the user or a TIE-In administrator modifies the user's information, including the user's profile, the user's application list, charge account numbers, etc.

Table 7. Description of TPM add request nametypes (Continued)

Nametype	Description
RESOURCE	sent when a TIE-In administrator creates an entry for an application or modifies an existing application's information, including server information, flags, charging information, etc.
REG_USER	sent when a user submits a registration for a TIE-In account or when a TIE-In administrator archives the user's registration
PROJECT	sent when a TIE-In administrator creates a project or modifies an existing project's information
SESSION	sent when a user logs into a TIE-In Front End System, thus initiating a session
APP_ACCESS	sent when a TIE-In administrator modifies the access control list for an application
APP_CONS	sent when a TIE-In administrator modifies the list of consultants for an application

For the format of the TIE_AUTH_REQ message, see Appendix A: Message Formats. The database tables are discussed in Appendix D: The TIE-In Database Tables. When a user's profile information is added or updated, the effected tables are *user_info*, *user_charge_accounts*, *user_applications*, and *user_gids*. For additions of and updates to an application's profile information, the effected tables are *application_info*, *application_costs*, *application_addresses*, and *application_groups*. When a user submits a registration or the registration is archived, the effected tables are *registration_info*, *registration_charge_info*, and *registration_logins*. The *current_sessions* table is written to when a session is initiated, while the *application_access* and *application_consultants* tables are written to when an application's access list or consultant list is modified, respectively.

TPM Information Request

Both the Session Manager and the Resource Request Manager obtain a variety of information from the database by sending AUTH_REQ_INFO type messages to the TPM. The information that can be obtained includes information about a user, an application, or a project.

The TPM determines which type of information is being requested by examining a field in the TIE_AUTH_REQ message called *nametype*. Table 7 describes the nametypes used in

association with an information request and for what purposes they are sent. In the actual request, the nametype is prefixed by TIE_AUTH_NAMETYPE_.

Table 8. Description of TPM information request nametypes

Nametype	Description
USER	sent when information about a user is requested
RESOURCE	sent when information about an application is requested
REG_USER	sent when a TIE-In administrator requests to view a user's registration
PROJECT	sent when information about a project is requested
PROJECTS	sent when a list of all projects is requested
SESSION	sent when information about the currently active sessions is requested (this information can only be viewed by TIE-In administrators)
APP_ACCESS	sent when the access list for an application is requested
APP_CONS	sent when the list of consultants for an application is requested
MF_USER	used to obtain a list of all registered users that match a specified flag value
NMF_USER	used to obtain a list of all registered users that do not match a specified flag value
MF_RESRC	used to obtain a list of all applications that match a specified flag
ADM_RESRC	used to obtain a list of applications for which a specified user is the application administrator
APP_USERS	used to obtain the list of users who are registered for the specified application and that are within the specified filter
CON_APPS	used to obtain the list of applications for which the specified user is a consultant

When the TPM receives a request for information, it determines which type of information is desired and calls a function in the database library to extract the information from the database. The information is then packaged into a TIE_AUTH_REPLY message and sent back to the caller. The format of the TIE_AUTH_REPLY message is discussed in Appendix A: Message Formats, and the functions in the database library are described in Appendix D: The TIE-In Database Tables.

TPM Delete Request

The Session Manager sends delete requests to the TPM to delete a user, an application, a project, or a user's registration. Only TIE-In administrators are authorized to submit this type of request to the TPM. Once again, the TPM determines which type of information is to be deleted by examining the value of the *nametype* field in the AUTH_REQ_DEL message. The different nametypes are shown and described in Table 7. In the actual request, the nametype is prefixed by TIE_AUTH_NAMETYPE_.

Table 9. Description of TPM delete request nametypes

Nametype	Description
USER	sent when a TIE-In administrator requests the deletion of a user's account
RESOURCE	sent when a TIE-In administrator requests the deletion of an application
REG_USER	sent when a TIE-In administrator successfully registers a user or requests the deletion of the user's registration
PROJECT	sent when a TIE-In administrator requests the deletion of a project
SESSION	sent when a user logs out of the TIE-In Front End System, thus terminating a session

When the TPM receives a AUTH_REQ_DEL message, it calls the appropriate function in the database library to delete the information from the database. The functions in the database library are discussed in Appendix D: The TIE-In Database Tables.

When a user's account is deleted, the affected database tables are *user_info*, *user_charge_accounts*, *user_applications*, and *user_gids*. When an application is deleted, the affected tables are *application_info, application_costs*, *application_addresses*, and *application_groups*. When a user's registration is deleted, the affected tables are *registration_info*, *registration_charge_info*, and *registration_logins*. Deleting a project affects the *project_access* and *project_info* tables, while deleting a session affects the *current_sessions* table.

TIE-In Application Server Process Description

The TIE-In Application Server (TAS) is the interface between the application's Graphical User Interface and the Resource Request Manager (RRM). Its function is to authenticate the received request, check any local authorization policy with regard to the user submitting the request, execute the application, and return status information on active and completed tasks.

The Front End System project will provide a TAS for a subset of hardware platforms (currently Sun, HP, and SGI). This server will be based on the model described here. An application may establish their own TAS, but they must conform to the protocol and message formats exchanged between the RRM and the TAS.

Communication with the RRM

The TAS listens for messages from the Resource Request Manager on IP/UDP port 1955, and sends replies back to port 1956 on the Common Services System. These messages, discussed earlier in the section "Resource Request Manager Process Description," are described in Appendix A: Message Formats and are labeled TIE_TIRRS_REQ and TIE_TIRRS_REPLY. When a request is received by the TAS, the message is decoded and the credentials are validated. If the credentials provided fail the validation process, then the TAS will generate a TIE_TIRRS_REPLY message. The status field will be set to RRM_AUTHFAIL and an appropriate description as to what the problem was will be placed in the info field.

The RRM sends six different types of messages to the TAS, as shown in Table 10. Each of these message types is discussed below.

Table 10. The Messages sent from the RRM to the TAS

Message Type	Message Description
TIRRS_SUBMIT_REQ	sent when a user runs the application
TIRRS_STATUS_REQ	sent when the RRM needs information about an active task
TIRRS_DEL_REQ	sent when a user wants to delete one or more of his active tasks, or when a TIE-In administrator wants to delete one or more active tasks

Table 10. The Messages sent from the RRM to the TAS (Continued)

Message Type	Message Description
TIRRS_ADD_ACCT	sent when a user attempts to add the application to his application list
TIRRS_DEL_ACCT	sent when a user removes the application from his application list
TIRRS_COMP_ACK	this message type is used by the RRM to notify the TAS that it has received the information about a task that has completed

The Configuration File

The TAS reads information about how to execute and manage an application from a configuration file that resides on the machine running the application. The name of the configuration file is */usr/etc/tiein.conf*, and it consists of 12 fields separated by colons. The first field contains the name of the application. When a request is received from the RRM, the TAS will search the configuration file for an application name that matches the name in the application field of the request. The configuration file is discussed in detail later in this section. (See "The Configuration File: /usr/etc/tiein.conf" under "TAS System Files.") The discussion below describes how the configuration file dictates the operation of the TAS for each of the message types.

Supported Authentication Levels

Regardless of the message type, the TAS verifies that the authentication level of the request meets or exceeds the level specified in the configuration file. The valid authentication levels are shown in Table 11. These levels can be logically ORed if more than one authentication level is supported. For example, if either Kerberos version 5 authentication or SecurID authentication is acceptable, the value for this field in the configuration file should be the logical OR of the hex values 2 and 4, which is 6.

Table 11. Valid Authentication Levels

Value	Authentication Level
2	Kerberos version 5 authentication is required
4	SecurId authentication is required

The authentication level is the level at which the user, or in some cases the RRM, has been authenticated. For message types TIRRS_COMP_ACK and TIRRS_STATUS_REQ, the

authentication level is that of the RRM. The RRM is authenticated using Kerberos Version 5. For these message types a test is done to ensure that the RRM has been authenticated at the Kerberos Version 5 level. This prevents unauthorized RRM processes from sending messages to the TAS. For all other message types, the authentication level contained in the message is that of the user submitting the request to run the application. In this case, the user's authentication level is checked against the supported authentication levels as specified in the configuration file.

Description of a Submit Request Message

This is the most common type of message sent from the RRM to the TAS, as it is sent whenever a user desires to run an application. This is also where most of the information contained in the configuration file is used.

The Local Authorization Policy

After validating the credentials and testing the minimum authentication level, the TAS will execute the local authorization policy specified in the configuration file for the application. If the field is empty, the TAS will assume no authorization is necessary and will continue processing the request. If the field is not empty, the TAS will execute the policy passing the arguments as shown in Table 12. If the policy returns a non-zero exit status, the TAS concludes that the user failed the local authorization policy and returns a reply message with a status of AUTHFAIL. If the policy returns an exit status of zero, the TAS concludes that the user successfully passed the authorization policy Using a local authorization policy allows access control beyond simple authentication. A basic authorization policy could simply check to see if the user's name is included on a list contained in a file. The complexity of the policy depends upon the needs of the application.

Table 12. The Arguments Passed to the Local Authorization Script

Argument Number	Argument
1	Name of the user submitting the request
2	Id of the user submitting the request
3	Address of the user submitting the request

Checking the Number of Current Tasks

Before running an application for an authorized user, the TAS checks to see if the maximum number of current tasks is already running. The maximum number of current

tasks can be specified in the configuration file. If running another task would exceed the specified maximum, the TAS returns a message to the RRM indicating that the task could not be run at the current time. If the application does not need to limit the number of current tasks, this field should be left blank in the configuration file. Note that it should not be set to 0 since this would essentially disable the application.

Creating the Status File

Next, the TAS will establish a status file by which the application may pass accounting and status information back to the TAS. The directory of this status file is taken from an entry in the configuration file. The format of the status file is discussed later in this section. (See "TAS Status file format" under "TAS System Files.") The value of this field in the configuration file can contain the string *$user*, which will be expanded to the username of the user submitting the task. This allows the status files for different users to be placed in different directories.

The pathname specified in the configuration file is appended with the filename *tie.task_id.tas_task_id*, where *task_id* is the task identifier issued by the RRM that was contained in the request message, and *tas_task_id* is assigned by the TAS, which in most cases will be the process id of the executed application.

If this field is left empty in the configuration file, the status file will be placed in the home directory. The home directory is also controlled via the configuration file and is discussed in the next paragraph. The name of the status file is passed to the application through the TIE_STATUS_FILE environment variable when the application is executed.

Setting the Home Directory

After creating the status file, the TAS will set the current working directory to the directory specified in the configuration file. This directory is know as the home directory. This field can contain any valid directory path on the local system. The string *$user* may also be used in the specification of the home directory. In such cases the user's username will be substituted for the string. For example, suppose the user "rdetry" submits a request and the home directory field contains the path */users/tiein/$user*. The TAS will substitute *rdetry* for *$user* and use */users/tiein/rdetry* as the home directory for tasks run by the user rdetry. In addition, the special string *~$user* can also be used within this field. This string will expand to the home directory defined for $user in the local system's /etc/passwd file. Status files will be placed in the home directory if a separate path is not specified in the configuration file for their placement.

Establishing the Environment

The next step the TAS performs is establishing the environment for the application. The TAS sets the environment variables shown in Table 13. In addition, a file may be specified in the configuration file that contains additional environment variables to be set by the TAS

prior to executing the application. Again, the string *$user* may be used in this file specification. The file itself should contain a list of environment variables, one per line, in the form:

NAME=VALUE

where NAME is the environment name such as **PATH** and VALUE is the value for that variable, such as **/bin:/usr/bin**.

Table 13. Environment Variables set by the TAS

Environment Variable Name	Environment Variable Value
DISPLAY	set to the Xserver display of the user
TIE_USER	set to the requesting user's username passed in the request
TIE_USERID	set to the string representation of the user's id passed in with the request
TIE_USER_ADDR	set to the users account IP address. This may not be the same as the IP address associated with the display.
TIE_USER_SSN	the social security number of the user. This will be empty if the user is not a US citizen.
TIE_USER_EMAIL_ADDR	the electronic mail address of the user
TIE_REQ_ID	set to the value of req_id passed with the request
TIE_TIRRS_REQ_ID	set to the tirrs_req_id which the TAS will assign. In most cases, this value will be the process id of the executed program.
TIE_XSERVER_KEY	(optional) set to the value of the MAGIC-COOKIE to be used in establishing windows onto the user's display
TIE_XFER_KEY	(optional-future)
TIE_BANDWIDTH	(optional) set to the maximum bandwidth of the user's network connection in bits per second
TIE_STATUS_FILE	filename to which the application should output status information
KRB5CCNAME	the name of the local credentials cache file when a user submits a request with forwardable credentials
XAUTHORITY	the name of the xauthority file which contains the magic cookie

Changing the Root Directory

The TAS offers applications the option of an additional level of security by allowing the specification in the configuration file of a new root directory. This feature can be used to prevent users from accessing the normal root directory. It is, however, somewhat difficult to use. If a new root directory is specified, all the commands and files required by the application must exist in the new root directory. In addition, the fields in the configuration file that specify the pathname of the application and the status file directory must be prefixed by the new directory. If a new root directory is specified in the configuration file, the TAS sets the new root directory by using the `chroot` command.

Setting the Process Owner

The TAS determines the process owner of the application by using the info in the configuration file. Valid values for the process owner include any user listed in */etc/passwd*, the string *$user*, or the string *$local_user*. If *$user* is used, the TAS will execute the application with the userid provided in the userid field of the request. If *$local_user* is used, the TAS will execute the application with the userid of the user contained in the password file on the local system. If any other username is specified, the TAS retrieves the userid of the user from the local */etc/passwd* file. The userid is then used to set the ownership of the application.

Executing the Application

Finally, the TAS will execute the application program as indicated in the configuration file. The application field should contain the full path name of the application program, along with any arguments to the application. Upon successful execution the TAS will place this task on an active task list, which it will maintain until the application terminates.

Monitoring the Status File

As mentioned before, the TAS uses a status file to communicate status and accounting information about a task to the RRM. The application can assume responsibility of the status file or leave this responsibility to the TAS. An entry in the configuration file is used to specify the *child type*. The value of this field can only be the string DIRECT_CHILD or the string INDIRECT_CHILD. In the first case, the TAS will assume the management responsibilities of the application. That is, after executing the application, the TAS will wait for it to complete. Upon completion, the TAS will update the status file, indicating that the application has completed. It will also write out the number of seconds that the application was running. This information is passed back to the RRM and on to the Accounting Manager. In the case of an INDIRECT_CHILD, the application itself is responsible for updating the status file. While this requires more work from the application developers, it also offers the opportunity for enhanced accounting procedures.

The format of the status file is such that up to seven accounting unit types may be set. The suggested unit types are

(1) Storage usage
(2) Memory usage
(3) CPU usage
(4) Connect time
(5) Technology Application definable
(6) Technology Application definable
(7) Technology Application definable

The application must fill in the desired fields in the status file with the appropriate values. If an application is not charging for a particular unit type, the field should be set to 0. The actual charge amounts are defined using the Front End System by a qualified application administrator. Thus the information in the status file is used, along with the charging information, by the Accounting Manager to determine the appropriate charges for the task.

Running as an INDIRECT_CHILD has the advantage that the status file can be updated while the task is still active. The TAS will detect when a status file has been modified, and it will then send a reply to the RRM with the latest information from the status file. A task running as a DIRECT_CHILD does not have this capability. In this case, the only time the status file is updated is upon completion of the task. It is up to the application to determine which child type is most desirable.

Description of a Status Request Message

The Resource Request Manager will periodically send messages to the TAS asking for the status of a particular task. After checking the authentication level of the RRM as described above, the TAS simply reads the status file that corresponds to the task specified by the RRM. It then determines the state of the task, either active or completed, and returns the state along with the latest accounting information to the RRM. The RRM then passes the accounting information to the Accounting Manager to store in the database.

Description of a Delete Task Request Message

The TIE-In Session Manager has an option that allows a user to delete one or more of her active tasks. It also allows a TIE-In administrator to delete one or more active tasks, regardless of the task owner. This is accomplished through the RRM. The RRM sends a delete task message to the TAS. The TAS verifies that the user requesting that a task be deleted is the owner of the task or a TIE-In administrator. If the user passes this test, the TAS sends a hang-up signal (SIGHUP) to the process id of the specified task. The process id of the task is obtained from the status file. Note that applications executed by the TAS must properly handle the hang-up signal. Also, if the application executed by the TAS starts another process, it is important that the new process id be written to the status file.

Description of an Add/Delete Account Request Message

When a user adds an application to his application list, the RRM sends an add account message to the TAS. Similarly, when a user removes an application from his list, the RRM sends a delete account message to the TAS. When the TAS receives this type of message, it checks the authentication level of the user as described above, and then executes the account script if one has been specified in the configuration file. This script can be used to set up or remove an account for a user. The arguments passed to the program or script are shown in Table 14. If the program or script returns a non-zero exit status, the TAS assumes the user's account was not successfully added or deleted. If a zero exit status is returned, the TAS will report back that the account was successfully added or deleted.

Table 14. The Arguments Passed to the add/delete Account Script

Argument Number	Argument Description
1	the string *add* for an add request, or the string *delete* for a delete request
2	the user's username
3	the user's userid
4	the user's group ids, in the form gid1,gid2,...,gidn or -1 if the user has no group ids
5	the user's social security number; may be empty if the user is not a US citizen
6	the user's directory to be mounted; may be empty if the user did not specify such a directory

Description of a Complete Acknowledgment Request Message

When a task completes, the TAS sets the status field of the reply message to indicate that the task has completed. When the RRM receives a reply for a task that has completed, it sends a complete acknowledgment message to the TAS. When the TAS receives this type of message, it removes the status file for the task. This is done so that the TAS does not remove the status file containing the final accounting information for a task until it is sure that the RRM has received this information. The TAS continues to send replies to the RRM

for a completed task until it receives a complete acknowledgment message or a large specified time-out period is exceeded.

Recovery of Active Tasks

Recovery of active tasks due to the termination of the TAS is achieved by maintaining a file copy of the active task list. Upon recovery of the TAS, the active task list can be re-established. The TAS will identify those processes that are still active, monitor the progress of those processes through their status files, and report back to the RRM.

TAS System Files

This section contains a description of the files used by the TAS and their formats.

The Configuration File: /usr/etc/tiein.conf

The configuration file is used by the TAS to retrieve the necessary information for it to execute and maintain an application. If the contents of any field require the use of the ":" character, then that character should be prefaced with the character "\". An entry may span multiple lines if the end of line is prefixed with the "\" character. White space may be present only at the beginning of a field. Comments can exist above or below an entry, but not within. All comments should be preceded by the # character. The entire entry must be no longer than 1024 characters. Fields may be empty; in those cases the word "(Optional)" will be stated in the field description. The format of an entry is as follows:

```
application_name: direct_child_flag: local_authorization_script: process_owner:\
home_directory:status_file_directory: environ_file: application args:\
add/delete script: supported_authentication_levels: change_root_directory:\
max_number_of_running_tasks
```

Field Description:

application_name: Name of the application. This name must match the application name provided in the TIE_TIRRS_REQ message.

direct_child_flag: Set to either DIRECT_CHILD or INDIRECT_CHILD. A direct child is managed by the TAS, which makes updates to the status file. An indirect child requires that the application make the necessary updates to the status file.

local_authorization_script: (Optional) The script or program to be executed that will perform any local authorization against the user or user's system. Arguments passed to the program will be user's

user name, user's numeric id, and the IP address of the user's system. If this field is empty then no local authorization will be performed.

Process_owner: The name of the owner of the application process to be executed by the TAS. The name given in this field will be looked up in the /etc/passwd file and the UID associated with that name will be the owner of the executed application. If the name given is the string *$user*, then the user's numeric id passed down with the request from the RRM will be used.

Home_directory: The full pathname of the directory that the TAS will set the current working directory to, prior to executing the application program. The string *$user* may be used in the description of the pathname, in which case the user's user name will be substituted. Additionally, the string *~$user* may be used. This string will be expanded to the home directory defined for $user in the local system's /etc/passwd file.

Status_file_directory: (Optional) The full pathname of the directory that will contain the status file for this task. The actual status file will have the name *tie.task_id.tas_task_id*, where *task_id* is the task identifier issued by the RRM that was contained in the request message, and *tas_task_id* is assigned by the TAS, which in most cases will be the process id of the executed application. The string *$user* may be used within the pathname. If this field is empty, the status files will be placed in the home directory.

Environ_file: (Optional) The full pathname of a file that contains environment variables to be established for this application. The information contained in this file is described above. The string *$user* may be used within the pathname, in which case the user's username will be substituted.

Application: The full pathname of the application, followed by any arguments, to be executed by the TAS.

Add/Delete script: (Optional) Program or script for adding and deleting accounts. See the description above for more information.

Supported Authentication levels: The level or levels of authentication required of a user in order to run the application. Values here can be 2 for Kerberos 5, 4 for SecurId, 8 for S/KEY, or a combination.

Chroot Directory:	(Optional) The directory to which the TAS will set the root directory (/) prior to executing the application.
Maximum number of running tasks:	(Optional) This value dictates the maximum number of tasks that can be simultaneously running. If no limit is to be imposed, this field should be left blank. If set to 0, no tasks will be run.

Examples:

```
# This is a valid configuration file, located in /usr/etc, named tiein.conf.
# The following is an entry for the resource named mpp_login.
mpp_login: \
        DIRECT_CHILD: \
        /users/tiein/bin/auth_script: \
        $user: \
        ~$user: \
        : \
        : \
        /usr/bin/X11/xterm: \
        /users/tiein/bin/add_user: \
        2: \
        : \
        25
#
# The following is an entry for the resource Xdemos.
#
Xdemos: \
        DIRECT_CHILD: \
        /tiein/config/authorize_user: \
        tiein_g: \
        /tiein/home: \
        /tiein/home: \
        /tiein/home/env_file: \
        /usr/bin/kerberos/xdemos: \
        /tiein/config/add_account: \
        2: \
        : \
        10
```

TAS Status file format

The status file will be located in the directory specified in the /usr/etc/tiein.conf file, with the name *tie.task_id.tas_task_id*, where *task_id* is the task id assigned by the RRM and *tas_task_id* is the task id assigned by the TAS. The status file is initially created by the TAS. In the case of a direct child, the TAS will make the updates to the status file. In the case of an indirect child, the application is responsible for the updates to the status file. The format of the status file is as follows:

```
completed(1/0):pid(optional):Completion message:units of unit type 1:\
units of unit type 2:units of unit type 3:units of unit type 4:units of unit type 5:\
units of unit type 6:units of unit type 7
```

Field Descriptions:

Completed(1/0):	Set to value "1" or "0". If task has completed, the value will be 1; otherwise the value is 0.
Pid:	(Optional) The process id to send the SIGHUP signal to when the user or administrator submits a delete task.
Completion message:	Brief message indicating the completion status of the application. If errors occur, this message will be used to inform the user as to the reason for the error.
Units of unit type [1-7]:	(Optional) The seven fields containing integers describing the number of units used by the task for unit types one through seven. If the fields are empty then the value is assumed to be 0. The suggested unit types are 1: Storage Usage 2: Memory Usage 3: CPU Usage 4: Connect Time 5: Technology Application definable 6: Technology Application definable 7: Technology Application definable

The following are all valid formats for a status file:

```
0:0:initial entry:0:0:0:0:0:0:0
0:12341:initial entry:::::::
0:6452:task is still active:0:0:0:7260:0:0:0
1:0:task has completed:0:0:0:126941:0:0:0
1:0:task has completed:145:16:523:290453:12:0:0
```

The Valid Front End Systems File: /usr/etc/tiein_fes

When the TAS receives a request, it also receives information about the Front End System that submitted the request. In order to verify that the request is from a valid Front End System, the TAS uses the file */usr/etc/tiein_fes*. This file simply contains a list of all of the valid Front End System names or IP addresses. When a request is received, the TAS attempts to match the front end system that sent the request with a name or an address in this file. If a match is found, the request is processed, otherwise the request is rejected.

Example:

The following is a valid /usr/etc/tiein_fes file. In this case, all three entries refer to the same machine.

```
#
# List of TIE-In Front-End System names: you can use addresses, primary names or
# aliases
#.....ONE PER LINE.....
#
134.218.37.137
tiein
sahp103
```

The Backup Directory: /usr/adm/tirrs_requests

This directory must exist for the TAS. This directory is used to maintain a file copy of the list of active tasks. This list is used to restore the active list in memory in case the TAS is shutdown while there are active tasks. This directory should be owned and modifiable only by root.

The Kerberos Key File: /usr/etc/tiein_v5srvtab

This file contains the keys used for Kerberos authentication of requests going to and from the RRM. When a request is received from the RRM, the TAS sends the credentials provided with the request, along with the key stored in this key file, to the Kerberos Authentication Service for authentication. If the authentication fails, the request is denied. This file should be owned and readable only by root. The key is stored as binary information, so it cannot be edited using a text editor. In order to change the key, the ksrvutil program must be used. This program is distributed along with the TAS executable file.

The Required IP Services: /etc/services

The following services need to be provided (added to the /etc/services file) for TIE-In.

```
tiein           1956/udp    # Communication with the FES
kerberos        88/udp      # Kerberos 5 authentication
kerberos_adm    751/tcp     # Kerberos 5 administration
```

The Kerberos Configuration File: /etc/krb.conf

The kerberos configuration file, */etc/krb.conf*, contains information used by the TAS and the *ksrvutil* program to communicate with TIE-In's Kerberos Authentication Server. This communication is necessary so that the TAS can perform authentication of requests received from the RRM. It is also used by the *ksrvutil* program to change the kerberos key in the local key file (*/usr/etc/tiein_v5srvtab*).

The kerberos configuration file will already exist if the machine serving the application is using the Kerberos Authentication System. If this is the case, the line shown below must be added to the file. If this file does not exist, it should be created and should contain the entry shown below. The required entry:

Sandia.EON.RSN tiein.sandia.gov admin server

At the present time, the only valid kerberos realm for TIE-In is Sandia.EON.RSN. In the future, cross-realm authentication will be supported, at which time a local Kerberos Authentication Service can be used.

The Kerberos Realms File: /etc/krb.realms

If the machine serving the application is already using the Kerberos Authentication System, a new file is required that defines in which realm the TIE-In Kerberos Authentication Service resides. The file is the kerberos realms file, */etc/krb.realms*. This file may already exist. If so, simply add the line shown below. If the file does not exist, create and add the line shown below. The required entry:

tiein.sandia.gov Sandia.EON.RSN

TIE-In Common File System

The TIE-In Common File System (CFS) provides a common file space to TIE-In users. It was designed to operate in a secure and seamless manner. The CFS is currently implemented using the Network File System (NFS) Version 2 and mount Version 1.

Motivation behind the CFS

A common file system is a necessary part of TIE-In for many reasons. First, users can run many different applications, each of which may require input files and generate one or more output files. Without a common file system, each of these applications will read and write files from and to the user's local home directory on the system that is serving the application. Thus, a user may end up with multiple files residing on multiple systems. With a common file system, all of the input and output files from all of the applications can be written to the user's common, network-mounted home directory.

Most TIE-In applications do not run as a process owned by the user, but as a process owned by a common user such as tiein. Thus, the CFS is required to perform the necessary translation between the user running the application and the user's username and userid on the remote file system. If this were not done, the application's attempt to read from or write to a mounted file system would fail.

NFS is generally viewed as being insecure and many routers block NFS and mount packets from systems outside of the local domain. The CFS improves the security of NFS and acts as a gateway so that NFS packets can be passed through a firewall. Instead of opening up the router to all mount and NFS packets from all remote systems, the router can be configured to allow these packets from the TIE-In Front End System only, resulting in a less vulnerable network configuration.

Another reason for the necessity of the CFS is to reduce the number of systems to which a file server must export a user's home directory. The CFS makes it possible to export the user's home directory to only one system (the TIE-In Front End System), as opposed to each of the systems on which the various applications of interest to the user are located.

Implementation

The Common File System consists of three processes:

(1) *mnt_auth* - performs authorization of the user and the system requesting to mount the user's home directory

(2) *mnt_server* - accepts and translates the remote procedure calls (RPC) issued by

the mount command on the application server and the responses by the mount daemon on the user's file server

(3) *nfs_server* - accepts and translates the RPCs issued by the application server and the NFS daemons on the user's file server.

In addition to these processes, the TIE-In Application Server (TAS) plays an important role in the Common File System. The TAS is a process that runs on the application server. The TAS issues the mount command, but before doing so it sends some authorization information to the *mnt_auth* process. The purposes of and the interaction between each of these processes will become clear as the operation of the Common File System is discussed.

The *mnt_auth*, *mnt_server*, and *nfs_server* processes need access to some common information about which systems have what directories mounted on whose behalf, and what translation needs to occur. This interprocess communication is implemented using shared memory.

The CFS Mount Process

There are four main steps required for an application to mount a user's home directory:

(1) authentication of the user

(2) initialization of the *mnt_auth* process

(3) authorization of the user and the application server

(4) mounting the directory

Each of these steps will now be discussed in more detail.

Authentication of the User

The first step is for the user to be successfully authenticated by the TIE-In Front End System. Authentication is used to verify the identity of the user and is accomplished using the Kerberos Authentication System and, if so specified, a SecurID hardware device. This is a simple but important step in the security of the CFS.

Initialization of the mnt_auth Process

Initialization of the *mnt_auth* process occurs when the user selects an application to run. If the application has been set up to mount the user's home directory (this is accomplished by a TIE-In administrator setting a flag via the TIE-In Session Manager), the Session Manager will send a message to the *xforward* process containing information about the user and the authorized application servers. The *xforward* process then sends this information to the *mnt_auth* process. This communication is shown in Figure 6.

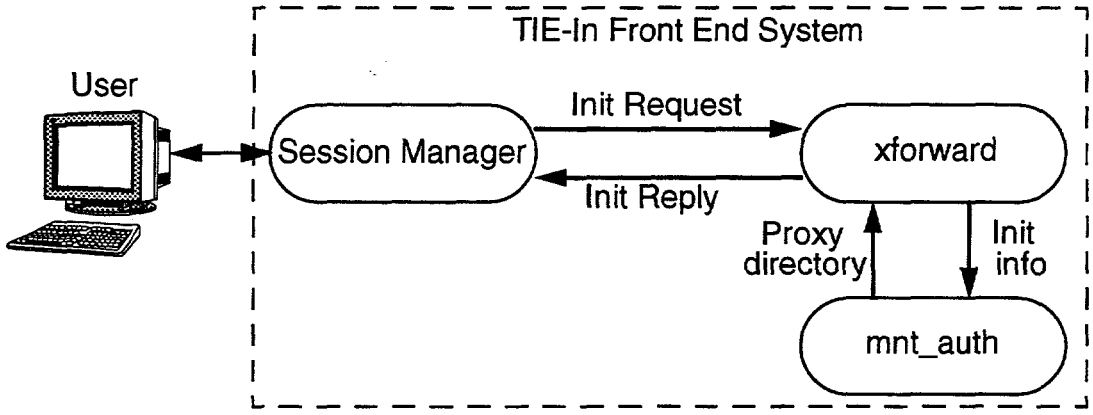


Figure 6. Initialization of the mnt_auth process.

The initialization request sent from the Session Manager to the *xforward* process contains Kerberos credentials for the user which are validated by *xforward* to ensure that the request was sent by an authenticated user running a valid Session Manager. The initialization information is then sent from *xforward* to *mnt_auth* via a UNIX socket connection, since these processes will always reside on the same system.

The initialization information is shown in Table 15. The information, with the exception of the list of authorized addresses, is sent as a single string separated by colons (:) which is parsed by *mnt_auth*. The *file server* and *directory* fields specify the user's home directory, which will be mounted by the *mnt_server* process. The *file_server_uid* is the user's TIE-In assigned user id and is used in the mapping from the process owner of the application to the file owner (the user). **It is therefore a requirement that the user id of the user on the file server is identical to his TIE-In user id.** If this were not the case, the mapping could not be done properly and the application would be denied access to the user's home directory.

Table 15. The initialization information sent to mnt_auth

Name	Description
file server	the name of the server on which the user's home directory resides
directory	the name of the user's home directory, located on the above server
application name	the name of the application that is mounting the user's home directory

Table 15. The initialization information sent to mnt_auth (Continued)

Name	Description
file_server_username	the user name of the user on the file server (this must match the user's TIE-In user name)
file_server_uid	the user id of the user on the server (this must match the user's TIE-In user id)
authorized addresses	the list of IP addresses that are authorized servers for the application

The list of authorized IP addresses is obtained from the information about this application that is stored in the TIE-In database. This list is used to ensure that a mount request is accepted only from an authorized system. The list is necessary because it is not yet known which of the potential servers will run the application.

The *application name* and the *file_server_username* fields are used by *mnt_auth* to generate a proxy directory, which is returned to the Session Manager. The Session Manager sends the proxy directory to the TAS with the submit request. The proxy directory will be used in the mount command executed by the TAS. The format of the proxy directory is *cfs_hostname:/application_name/file_server_username,* where *cfs_hostname* is the fully-qualified name of the machine on which the CFS processes are running. The proxy is necessary because the application server will actually mount the proxy directory, which results in the mount command being sent to the CFS. When a mount RPC is received, the CFS uses the proxy directory as an index into a table from which the user's actual file server and directory are obtained and subsequently mounted.

During this step, *mnt_auth* generates a timestamp that will be checked when the authorization information is received from the TAS. If the time at which the authorization request is received is not within the specified time window, the authorization will fail. The timestamp is also used to remove stale entries from the shared memory table. An entry can become stale if the TAS process is not running on the application server. In this case, an authorization request is never sent from the TAS to *mnt_auth*, the timestamp expires, and the information is removed when another initialization request is received.

Authorization of the User and the Application Server

Prior to performing the mount, the TAS sends some information to *mnt_auth* for authorization. The information is sent via a TCP socket and contains the user's Kerberos credentials that are validated by *mnt_auth* to ensure that the request originated from an authenticated user and a valid TAS process. Figure 7 shows the communication between the TAS and *mnt_auth*, and Table 16 shows the information that is sent from the TAS to *mnt_auth*.

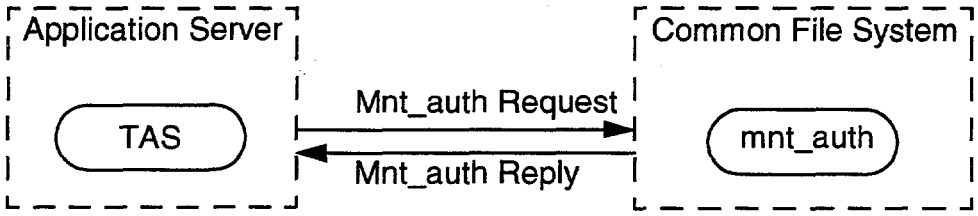


Figure 7. Authorization of the user and the application server.

The proxy directory is used by *mnt_auth* as an index into a table in which the initial information sent by the Session Manager is stored. If the proxy directory does not exactly match the one generated by *mnt_auth* in the initialization step, the table lookup will fail and this in turn causes the authorization to fail. From the table, *mnt_auth* obtains the username and user id of the user on the user's file system (*file_server_username* and *file_server_uid*), as well as the list of authorized system addresses. The username and user id are verified against the *file_server_username* and *file_server_uid* fields in the request. If either does not match, the authorization fails. In addition, the *file_server_username* is compared to the username in the Kerberos credentials contained in the request. If it does not match, the authorization fails. This test ensures that the user has been authenticated.

If the above tests are successful, the address of the system that sent the request is then checked against the list of authorized system addresses. If it is not on the list, the authorization fails. In addition, the timestamp set during initialization is checked to verify that the authorization request has been received within the specified time window. If the request was received outside of the window, the authorization fails. Otherwise, the timestamp is reset to the current time so that the timing can be verified when the mount request is received from the application server.

Table 16. The information sent by the TAS to mnt_auth

Field	Description
proxy_directory	the proxy directory generated by mnt_auth in the initialization step
app_owner_name	the username of the owner of the application process on the application server
app_owner_uid	the user id of the owner of the application process on the application server

Table 16. The information sent by the TAS to mnt_auth (Continued)

Field	Description
file_server_username	the user_name of the user on the file server (must match the user's TIE-In username)
file_server_uid	the user id of the user on the file server (must match the user's TIE-In userid)
app_server_address	the IP address of the application server

If all of the tests are successful, *mnt_auth* stores the *app_owner_name*, *app_owner_uid,* and the address of the application server in the table. The *app_owner_name* field contains the name of the user that is the owner of the application when it is executed. Similarly, the *app_owner_uid* field contains the userid of the owner of the application. This information is essential for the operation of the *nfs_server*. The *nfs_server* maps the *app_owner_uid* to the *file_server_uid* so the permissions will be correct in the user's home directory.

Mounting the Directory

If the authorization in the previous step is successful, the TAS then mounts the proxy directory (via the execvp command). The *mnt_server* process of the CFS takes the place of the normal mount daemon. It registers with the portmapper and looks identical to a mount daemon to the application server. The communication between the application server, the *mnt_server* process, and the file server is shown in Figure 8.

When the *mnt_server* receives a mount RPC, the first thing it does is read information from the table using the proxy directory as an index. This information includes the file server and the user's directory on the file server. It then verifies that the system that sent the mount is the system that was authorized by *mnt_auth* in the previous step. If it is not, an error message is logged and the mount fails. Otherwise, *mnt_server* then checks the timestamp to verify that the mount was received within the allowable time window. If it was not, an error message is logged and the mount fails.

The *mnt_server* process then determines if this server/directory combination has already been mounted. If it has not, *mnt_server* writes some additional information to the shared memory table for this proxy directory. It then issues a mount RPC to the user's file server, requesting to mount the user's home directory. If the mount is successful, the file handle of the directory is returned to *mnt_server*. This file handle is stored in the table and a proxy file handle is created. The proxy file handle contains indices into the shared memory table. This makes the operation of the *nfs_server* faster, since it can quickly determine the server/directory combination to which to send the NFS RPC. The proxy file handle is then returned to the application server, and the mount successfully terminates.

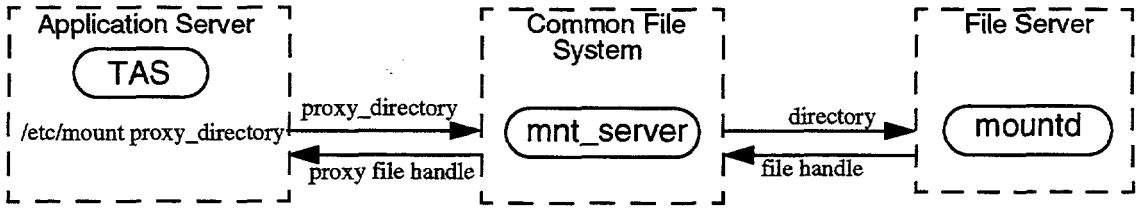


Figure 8. Mounting the user's home directory.

If the server/directory combination has already been mounted by *mnt_server* when the mount RPC is received from the application server, the application server is added to the list of systems that have this server/directory mounted. In this case the proxy file handle that is already stored in the shared memory table is returned. The server/directory combination will have already been mounted if the user is already running the application on another application server or another application that mounts the user's home directory.

The Mount Point

The user's home directory is mounted on the directory *tie_home* underneath the local home directory specified for the user in the application's entry in the TIE-In configuration file (/usr/etc/tiein.conf) on the application server. It is not mounted directly over the local home directory specified for the user in case the application wants to have some setup or configuration files in the user's area, but not on a remote system. This may be done for security or for performance reasons. It also allows the application to write intermediate files to the user's local home directory, and then move the final output to the user's remote home directory upon completion of the application.

Figure 9 shows the directory structure described above. The TAS executes the application from the directory specified as the home directory for the user. This is shown in the figure as TIE_HOME. The user's remote home directory is mounted on the tie_home directory underneath TIE_HOME.

Mounting on an Add or Delete Request

The previous section described how the CFS mounts a user's home directory when the user runs an application. The CFS also mounts the user's home directory when he adds the application to his application list and when he removes the application from his application list. The mount is performed in these cases because the application may have some configuration or default files that need to be placed in the user's directory when he first adds the application to his list. Similarly, if the user removes the application from his list, the application may need to remove some files from the user's home directory.

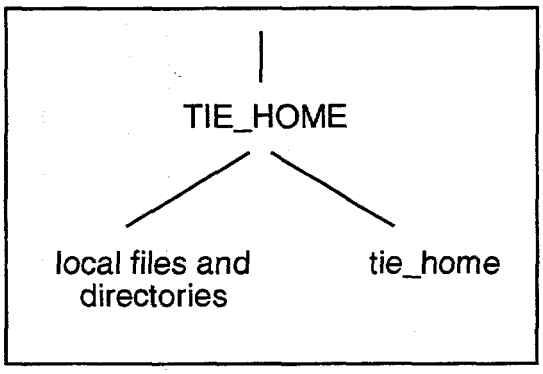


Figure 9. Illustration of the tie_home mount point.

When the user attempts to add an application to his list, he does not yet have a home directory on the application server. Therefore, the CFS cannot mount his home directory on a remote system to the *tie_home* directory underneath his local home directory as is done when the user runs an application. Therefore, his home directory on the remote system is mounted on a temporary mount point. The name of the temporary mount point is passed by the TAS to the application's add/delete account script as the seventh argument. Thus, if the script needs access to the user's remote directory, it can be accessed via the temporary mount point. When the add/delete account script terminates, the user's home directory is unmounted.

Operation of the NFS Server

The NFS server process (*nfs_server*) accepts NFS RPCs from the application server, performs some necessary translations, and sends an NFS RPC to the file server (see Figure 10). The information contained in the RPC depends upon which NFS procedure is to be called on the file server, but in general a file handle for the file of interest is provided. The file handle is a proxy file handle that is used by *nfs_server* to quickly access the file server's information in the shared memory table.

The first step performed by *nfs_server* is to verify that the system sending the RPC is one of those on the list of systems that have been previously authorized and have successfully mounted the corresponding proxy directory. If the system fails this test, an error message is logged and RPC_AUTHERR is returned. The *nfs_server* process also verifies that the user id in the RPC matches the *app_owner_uid*. If there is a mismatch, an error message is logged and RPC_AUTHERR is returned.

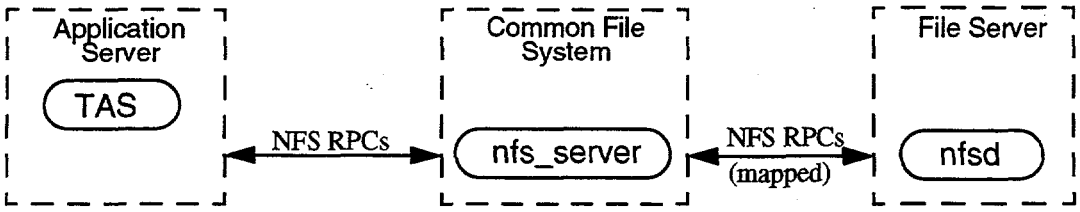


Figure 10. Operation of the nfs_server.

If the RPC passes these tests, *nfs_server* then determines the file server to which this RPC is to be sent, and retrieves the correct file handle to be used. Depending upon the NFS procedure being called, the uid in the request may have to be translated to the uid of the user on the file server (from *app_owner_uid* to *file_server_uid*). This translation is performed if necessary, and the RPC is sent to the file server. When a response is received, the uid may have to be translated back to the uid of the owner of the application on the application server (from *file_server_uid* back to *app_owner_uid*). This translation is performed if necessary, and the results of the RPC to the file server are returned to the application server that initiated the request. For more information about the NFS procedures, see the document *Network File System: Version 2 Protocol Specification*, published by Sun Microsystems.[2]

As with the nfsd process, many *nfs_server* processes can be running simultaneously to achieve greater response. The default number of *nfs_server* processes is four, but this can be changed easily at run time.

Unmounting a File System

Prior to executing an unmount, the TAS will send an authorization message to the *mnt_auth* process, informing it that a umount will soon be executed. Included in the authorization message is the proxy directory that is to be unmounted. The *mnt_auth* process looks up the table entry for this proxy directory and verifies that the system that sent the authorization message is one that successfully mounted the proxy directory. If it is, *mnt_auth* sets a flag indicating that a umount RPC is to be expected soon. It also sets a timestamp, which is checked by *mnt_server* when the umount RPC is received.

The TAS issues a umount command (via the execvp command) when an application completes, or in that case of an add or delete account request, after the add/delete account script has completed. If the user is running the application more than once, the umount is sent only at the completion of the last running application.

When the umount RPC is received by *mnt_server*, it first verifies that the RPC was sent by an authorized system. It then checks the mount flag that will have been set by *mnt_auth* to indicate that an unmount is pending, and verifies that the unmount has been received within the allowable time window. If both of these tests are successful, *mnt_server* removes the

table entry for this proxy directory. It then checks to see if this is the last application server that has this server/directory combination mounted. If it is, it sends a umount RPC to the server requesting to unmount the user's home directory. If this is not the last system to have this server/directory combination mounted, it simply returns an RPC to the calling system indicating that it has successfully unmounted the proxy directory.

Security Tests

A number of tests were performed to ensure that the TIE-In File System is secure. The greatest potential threats that were identified include (1) an unauthorized system being able to mount the user's home directory, and (2) an unauthorized system or user being able to intercept and reissue the NFS RPCs. Since the TIE-In CFS is based upon NFS, some of the security problems present in NFS are also present in CFS. The greatest threat is the lack of a strong authentication of the RPCs. The authentication level used is AUTH_UNIX, which is simply a username and userid. This authentication mechanism can potentially be defeated by sending low-level RPCs containing another user's username and userid to the file server. The CFS helps to eliminate this threat by ensuring that the RPCs are coming from the system that performed the mount. Thus, only another user with root access on the application server could be a threat, and this type of threat is virtually impossible to eliminate.

Mount Authentication Test

Two tests were performed to ensure that an unauthorized system cannot mount a user's home directory. For the first test, the TIE-In Application Server process running on an authorized system (sahp165) was stopped. The application was then executed, and an attempt was made to mount the proxy directory, created for the user by the CFS, from an unauthorized system (sass498). The attempt failed. The message returned to sass498 was that the server was busy. A message appeared in the TIE-In log file indicating that an unauthorized system attempted to mount the proxy directory of the user. The message includes the IP address of the unauthorized system. A mount request was also issued from the authorized system (sahp165), but this was denied as well. The reason for this is that an additional authorization message must be sent from the TAS prior to the mount request being sent. Since the TAS was not running, the authorization message was not sent so the mount request was denied.

For the second test, the TAS was modified to sleep for 60 seconds in between sending the mount authorization message and performing the mount. In addition, the time window between receiving the authorization request and the mount request was increased to 70 seconds. While the TAS was sleeping, a mount request was sent from an unauthorized system (sass498). Again, the mount request failed with the message server busy and an error message was logged. When the mount request was issued from the authorized system, the mount was successful. This was expected, because once the mount authorization message has been successfully sent, the system is authorized to perform a mount.

NFS Authentication Test

A test was performed to ensure that only authorized systems can successfully send NFS remote procedure calls (RPC) to the CFS. A simple NFS client was created that sends a few basic RPCs to the CFS. The client was run from a number of different systems at different times to verify that only authorized systems can successfully send RPCs to the CFS. An authorized system is one that has successfully mounted a file system on behalf of the user through the CFS, and on which the application is running. For example, if user rdetry is running the application Xdemos on sahp103, sahp103 is an authorized system. When the NFS test client was run on sahp103, the RPCs were successful. It is important to note, however, that this required the knowledge of the proxy file handle, which was determined from examining the code. Normally, the only way a file handle can be obtained is to snoop the network.

The test was repeated when no applications were running on sahp103, and the RPCs failed with the message of "Permission Denied." The same result occurred when the test was run from sahp165 when an application was running on sahp103. The same result also occurred when user gmachin ran the test while user rdetry was running an application on sahp103. Thus, not only is the client's address checked, but so is the uid of the user.

The only threat is again from a knowledgeable user with root access on a system running an application. This type of user would also be able to cd to the user's mounted file system anyway, and therefore would never go through the trouble of creating a sophisticated NFS client.

In order to successfully attack, the attacker would have to (1) get the root file handle, or the file handle of the file of interest, (2) either execute the attack from the machine running the application, or spoof as this machine, (3) obtain the uid of the process owner of the application and set his credentials to this uid, and (4) have a detailed knowledge of NFS RPCs and write a program to do the attacking.

Special Considerations for Application Developers

This section contains some information for application developers and maintainers. Most of the information appears in other sections of this document, but it is repeated here for easy reference.

The first step in setting up an application to mount a user's remote file system is to modify the flags in the applications entry in the TIE-In database. Only a TIE-In administrator can make this modification. Once the modification has been made, the Front End System will send the information required by the TAS to mount the user's remote home directory when the user runs the application, or requests that the application be added to or deleted from his application list. This same process can be used to modify the application's database entry if the application no longer wants or needs to mount the user's remote directory.

When the user runs an application, his remote home directory is mounted on the *tie_home* directory underneath his local home directory. The user's local home directory is specified in the TIE-In configuration file (/usr/etc/tiein.conf) and is typically created when the user requests the addition of the application to his application list. Since the user's local home directory does not exist when he requests the addition of the application to his list, the user's remote home directory is mounted on a temporary mount point. The mount point is passed to the add/delete account script as the seventh argument. When the add/delete account script terminates, the remote directory is unmounted.

To achieve the best performance and reliability, the application might want to read and write all intermediate files to and from the user's local home directory. Files can be copied to or from the remote home directory at the start and completion of the application. This helps to ensure that important information is not lost if the connection to the file server is lost. It will also help to achieve greater performance since reads and writes to a local file system are faster than those to a mounted file system.

Specifying the User's Home Directory

In the current implementation of the TIE-In Front End System, the user's home directory must be set by a TIE-In administrator. It cannot be set by the user. In the future, the user may be able to specify a home directory for each of the applications on his list, but the security implications of this have not yet been thoroughly investigated.

The user's home directory can be a file system on his workstation's local disk, or a file system created for the user on a TIE-In file server. The latter case will probably be more common since the first case requires the ability of the application server to mount the user's local disk. This is unlikely due to the increasing presence of firewalls. In addition, for external users it is likely that the network bandwidth between an application server and a TIE-In file server will be better than the bandwidth between the application server and a user's local disk. Internal Sandia users, however, may find it useful to mount a local disk.

Performance

One of the main requirements in the design of the CFS was to avoid dramatically slowing down the performance of NFS. This was achieved by using shared memory for the interprocess communication, and designing the tables in such a manner so that they could be rapidly accessed. Some slowdown was unavoidable of course, since twice as much network communication must take place, as well as security tests and mappings of the user id and server address.

Some tests were performed to determine the overhead of going through the CFS as opposed to using NFS directly, and the results are shown in Tables 17 through 22. In Test 1, shown in Table 17, the read performance was tested. A 1.25 Mbyte file was copied from a mounted file system to a local file system using NFS alone and while using CFS. In this case the

application was running on an HP Workstation (sahp165), and the home directory was on a Sun Solaris workstation (sass498). For all tests the CFS was running on another HP workstation (sahp103).

Table 17. Results of CFS vs. NFS read performance: Test 1

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	15.0	11.6	29.30
4	12.5	11.6	5.04
6	12.8	11.9	7.56
8	12.1	11.9	1.68
10	12.2	11.6	5.17

In test 2, shown in Table 18, the same test was performed except the application was running on a Sun Solaris workstation (sass498) and the home directory was on an HP workstation (sahp165).

Table 18. Results of CFS vs. NFS read performance: Test 2

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	11.1	11.1	0.00
4	12.5	11.4	7.89
6	11.1	11.1	0.0
8	11	11	0.0
10	11.1	11.1	0.0

In test 3, shown in Table 19, the write performances were compared. The setup is the same as in test 1, but in this case the 1.25 Mbyte file was copied from the local disk to the mounted file system.

Table 19. Results of CFS vs. NFS write performance: Test 3

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	45.6	12	280.0
4	12.1	11.9	2.54

Table 19. Results of CFS vs. NFS write performance: Test 3 (Continued)

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
6	11.9	11.9	0.0
8	11.9	11.9	0.0
10	12.1	12	0.83

In test 4, shown in Table 20, the write performance was repeated using the same system configuration as in test 2.

Table 20. Results of CFS vs. NFS write performance: Test 4

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	11.5	11.3	1.77
4	12	11	9.09
6	12	11	9.09
8	12.3	11.1	10.81
10	12	11	9.09

In test 5, shown in Table 21, the read/write performance of the CFS was compared to that of NFS alone. The system configuration was the same as in test 1, but in this case the 1.25 Mbyte file was copied from the mounted file system to another file on the mounted file system.

Table 21. Results of CFS vs. NFS write performance: Test 5

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	47.2	21.6	118.52
4	26.2	21.3	23.00
6	26.7	21.5	24.12
8	27.4	21.5	27.44
10	27.4	21.4	28.04

In test 6, shown in Table 22, test 5 was repeated using the system configuration described in test 2.

Table 22. Results of CFS vs. NFS write performance: Test 6

# of nfs_servers	CFS Avg. time (secs)	NFS Avg. time (secs)	Overhead (%)
1	20	15	25.00
4	18.7	14.9	20.03
6	17.4	14.8	17.57
8	16.8	15.1	10.76
10	17.2	15.5	10.97

The results of the performance test indicate that the overhead is quite low (0 to 10 percent) for straight reading and writing, but increases significantly for mixed reading and writing (10 to 25 percent). The operation of most TIE-In applications is such that mixed reading and writing will rarely occur. In the majority of cases, files will be loaded from the user's home directory, some processing will occur, and an output file will be written back to the user's home directory. In these cases, the performance of the CFS is such that the slowdown will be negligible. Even a slowdown of 25 percent is quite acceptable, given the fact that without the CFS, accessing the user's home directory is not possible at all.

Future Work

The performance of the CFS will be closely monitored and improvements will be made when possible. Since it is difficult to determine just how the CFS will be used and how it will perform during testing, it was designed so that many parameters can be set at run time. Different combinations of the parameters will result in different performances.

The current implementation of the CFS uses NFS Version 3 and mount Version 1. In the future, the CFS will have to support NFS Version 3 and mount Version 2. In addition, if the Distributed Computing Environment (DCE) becomes widely used, the CFS will also have to support DCE's Distributed File System (DFS).

Intentionally Left Blank

Interprocess Communication

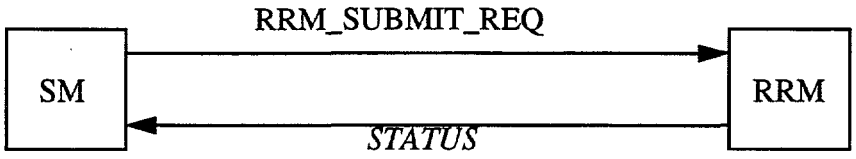
This section describes the communication that takes place between the various Front End System processes, including the Resource Request Manager (RRM), the TIE-In Profile Manager (TPM), the Accounting Manager (AcctM), the Session Manager (SM), and the TIE-In Application Server (TAS). The format of the messages are discussed in Appendix A: Message Formats.

Communication Between the SM and the RRM

The following describes the exchange that takes place between the SM and the RRM for the various types of request messages, and possible reply messages for those requests. In the diagrams below, flow towards the right indicates a request submitted by the SM to the RRM, while flow towards the left indicates a reply from the RRM sent to the SM. The labels on the arrows indicate the request or reply type, while *STATUS* indicates the status of the reply.

Submit Request

When a user selects an application from her application list, the SM sends a submit request to the RRM. The RRM then reformats the request and sends it to the TAS that is serving the selected application (see "Submit Request" in the "Resource Request Manager Process Description" section). The request that is sent from the SM to the RRM is described below.



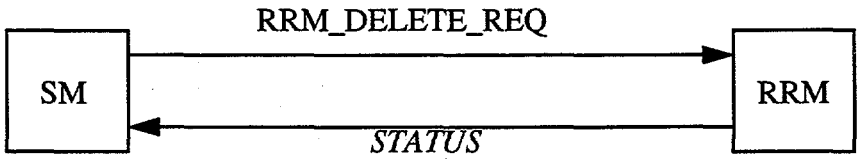
where *STATUS* is one of the following:

RRM_SUCCESS:	The request was successfully submitted to the TAS and the user is now running the application.
RRM_AUTHFAIL:	The user failed the authentication policy of the application.
RRM_XFAIL:	The application could not establish a connection with the user's workstation or PC.
RRM_RESRC_UNAVAIL:	The application is currently unavailable.

RRM_REQ_EXISTS:	A task with the task id assigned by the RRM for the submitted task is already running. This can occur when communication between the RRM and the TAS is slow. The RRM sends the submit request to the TAS multiple times before giving up. Sometimes the TAS has successfully processed the request but the RRM has not received the reply before sending out another submit request. When the new submit request arrives, the task is already active.
RRM_RESRC_UNK:	The application is unknown by the TAS. This typically occurs when there is an error in the TAS configuration file.

Delete Task Request

The SM has an option that allows the user to delete one or more of her active tasks. This option can also be used by a TIE-In administrator to delete a task regardless of who owns the task. When this option is selected, the SM sends a delete request to the RRM. The RRM reformats the request and sends it to the TAS that is serving the application (see "Delete Task Request" in the "Resource Request Manager Process Description" section). The request sent from the SM to the RRM is described below.



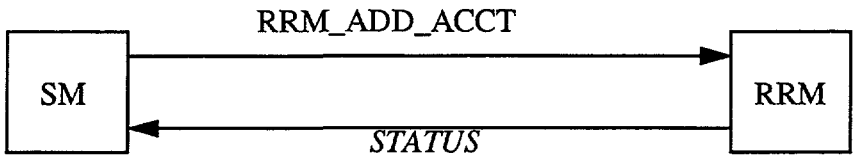
where *STATUS* is one of the following:

RRM_SUCCESS:	The specified task was successfully deleted.
RRM_AUTHFAIL:	The user is not the owner of the task and not a TIE-In administrator.
RRM_REQUNK:	The specified task is not known by the TAS. This can occur when a task completes just prior to the TAS receiving the delete request.
RRM_RESRC_UNK:	The application is not known by the RRM or the TAS, or the TAS encountered an error while reading the status file for the task.

Add Account Request

When a user attempts to add an application to her application list, the SM will send an add account request to the RRM. The RRM reformats the request and sends it to the TAS (see "Add Account Request" in the "Resource Request Manager Process Description" section).

The TAS will execute the add account script if one was provided for the application. The add account request sent from the SM to the RRM is described below.

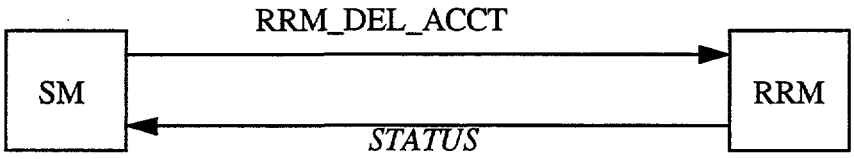


where *STATUS* is one of the following:

RRM_SUCCESS:	The account was successfully added.
RRM_AUTHFAIL:	The request failed authentication or authorization policies.
RRM_RESRC_UNAVAIL:	The application is currently unavailable.
RRM_RESRC_UNK:	The application requested is unknown by the TAS.

Delete Account Request

When a user attempts to delete an application from her application list, the SM will send a delete account request to the RRM. The RRM reformats the request and sends it to the TAS (see "Delete Account Request" in the "Resource Request Manager Process Description" section). The TAS will execute the delete account script if one was provided for the application. The delete account request sent from the SM to the RRM is described below.



where *STATUS* is one of the following:

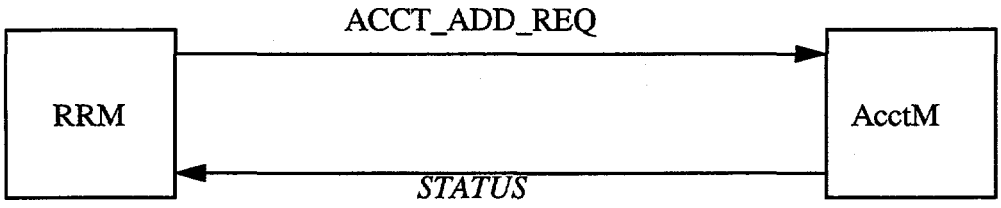
RRM_SUCCESS:	The account was successfully deleted.
RRM_AUTHFAIL:	The request failed authentication or authorization policies.
RRM_RESRC_UNAVAIL:	The application is currently unavailable.
RRM_RESRC_UNK:	The application requested is unknown by the TAS.

Communication between the AcctM and the SM and RRM

This section discusses the communication that takes place between the AcctM and the SM and RRM for the various messages, and shows the possible replies.

Add Request

The RRM sends an add request to the AcctM when a user successfully submits a task. The AcctM creates an entry in the database for the task.

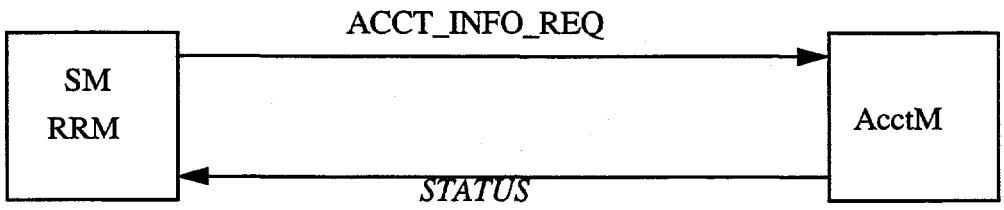


where *STATUS* is one of the following:

ACCT_REPLY_SUCCESS:	The request was successfully processed and the information about the task has been stored in the database.
ACCT_REPLY_FAILED:	The request failed and an error message is provided in the reply.

Information Request

The RRM and SM will send information requests to the AcctM to obtain information about a task or a group of tasks matching a set of criteria.

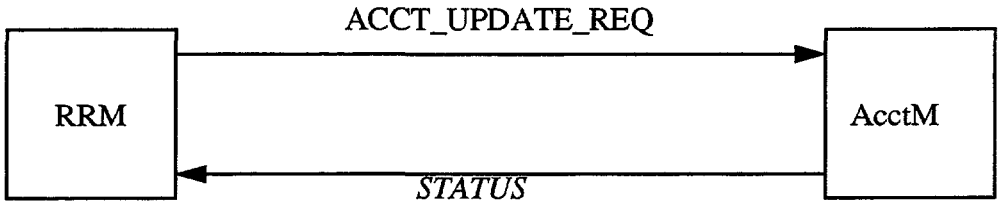


where *STATUS* is one of the following:

ACCT_REPLY_SUCCESS:	The request was successfully processed and the reply contains the desired information.
ACCT_REPLY_NOENTITY:	No tasks were found that match the specified search conditions.
ACCT_REPLY_FAILED:	The request failed and an error message is provided with the reply.

Update Request

When the RRM receives an update from the TAS concerning a task, it will send the information to the AcctM via an update request.



where *STATUS* is one of the following:

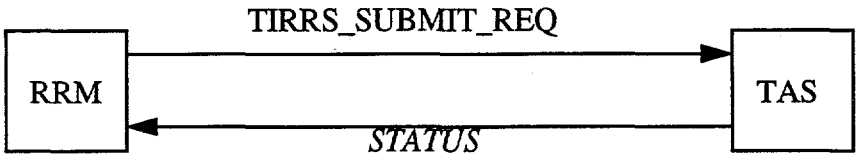
ACCT_REPLY_SUCCESS:	The request was successfully processed and the information about the task has been updated.
ACCT_REPLY_NOENTITY:	No requests were found that match the specified search conditions.
ACCT_REPLY_FAILED:	The request failed and an error message is provided with the reply.

Communication between the RRM and a TAS

The following describes the exchange that takes place between the RRM and the TAS for the various types of request messages, and possible reply messages for those requests. Flow towards the right indicates a request submitted by the RRM to the TAS, while flow towards the left indicates a reply from the TAS sent to the RRM. The labels on the arrows indicate the request or reply type, while *STATUS* indicates the status of the reply.

Submit Request

The RRM will send a submit request to the TAS when a user selects an application from his application list.

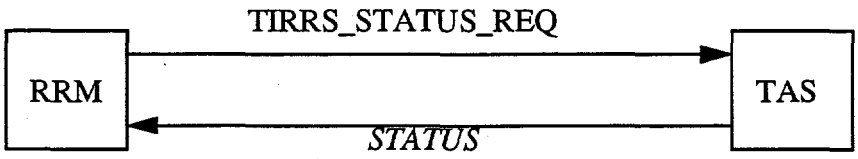


where *STATUS* is one of the following:

TIRRS_RRM_SUCCESS:	The task was successfully submitted and the application is now in control.
TIRRS_RRM_AUTHFAIL:	The request failed authentication or authorization policies.
TIRRS_RRM_XFAIL:	Could not open display to users X server, request was not submitted.
TIRRS_RRM_REQ_EXISTS:	The task already exists and is executing. The reply will contain the appropriate tas_task_id for the previously submitted task.
TIRRS_RRM_RESRC_UNAVAIL:	The task cannot be submitted due to lack of resources. See info for further detail.
TIRRS_RRM_RESRC_UNK:	The application requested is unknown by the TAS.

Status Request

The RRM will send a status request to the TAS at periodic intervals to determine the state of a task.



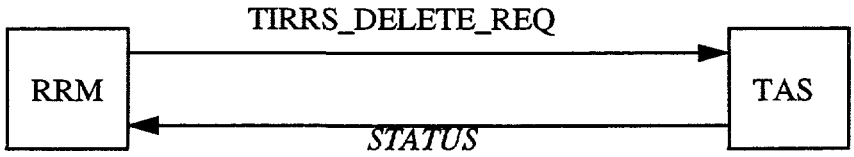
where *STATUS* is one of the following:

TIRRS_RRM_REQ_UPDATE:	The task is currently running and the *account_info* field of the reply contains the current information, if available.
TIRRS_RRM_AUTHFAIL:	The request failed authentication or authorization policies.
TIRRS_RRM_RESRC_UNAVAIL:	The request cannot be fulfilled due to lack of resources. See info for further detail.
TIRRS_RRM_RESRC_UNK:	The application requested is unknown by the TAS.

TIRRS_RRM_REQUNK:	The specified task is not known on this system. It may be that the task completed some time ago and the TAS no longer has a record of its completion.
TIRRS_RRM_REQCOMPL:	The task has completed and the *account_info* field of the reply contains the final accounting information.

Delete Request

The RRM sends a delete request to the TAS when a user wishes to delete a current task.

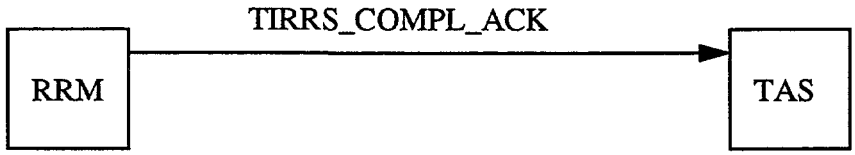


where *STATUS* is one of the following:

TIRRS_RRM_SUCCESS:	The task was successfully deleted.
TIRRS_RRM_AUTHFAIL:	The request failed authentication or authorization policies.
TIRRS_RRM_RESRC_UNAVAIL:	The request cannot be fulfilled due to lack of resources. See info for further detail.
TIRRS_RRM_RESRC_UNK:	The application requested is unknown by the TAS.
TIRRS_RRM_REQUNK:	The specified task is not known on this system. It may be that the task completed some time ago and the TAS no longer has a record of its completion.
TIRRS_RRM_REQCOMPL:	The task has completed and the *account_info* field of the reply contains the final accounting information.

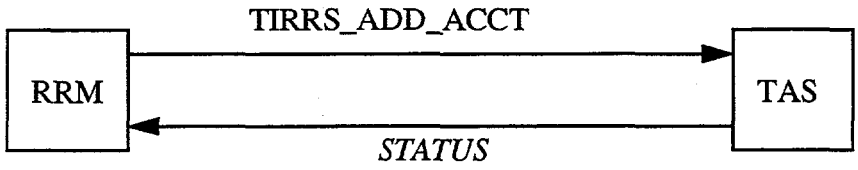
Complete Acknowledgment

The request type TIRRS_COMPL_ACK is an acknowledgment by the RRM to the TAS that the task completed message was received. No response from the TAS is expected. The TAS will keep the task on its active task list until a completed acknowledgment is received.



Add Account Request

The RRM will send an add account request to the TAS when a user attempts to add a new application to her application list.

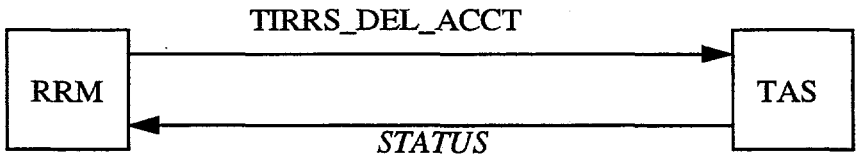


where *STATUS* is one of the following:

TIRRS_RRM_SUCCESS:	The account was successfully added.
TIRRS_RRM_AUTHFAIL:	The request failed authentication or authorization policies.
TIRRS_RRM_RESRC_UNAVAIL:	The request cannot be fulfilled due to lack of resources. See info for further detail.
TIRRS_RRM_RESRC_UNK:	The application requested is unknown by the TAS.

Delete Account Request

The RRM will send a delete account request to the TAS when a user removes an application from her application list.



where *STATUS* is one of the following:

TIRRS_RRM_SUCCESS:	The account was successfully deleted.
TIRRS_RRM_AUTHFAIL:	The request failed authentication or authorization policies.
TIRRS_RRM_RESRC_UNAVAIL:	The request cannot be fulfilled due to lack of resources. See info for further detail.
TIRRS_RRM_RESRC_UNK:	The application requested is unknown by the TAS.

TAS Status Reply

The TAS will periodically send uninitiated status replies to the RRM. The accounting information supplied with the reply is the latest available information and is passed on to the AcctM.

The update will have a status of

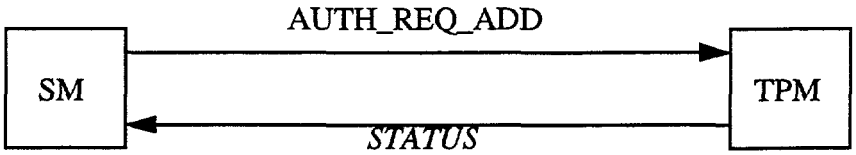
TIRRS_RRM_REQ_UPDATE:	The task is currently running and the *account_info* field of the reply contains current information.
TIRRS_RRM_REQCOMP:	The task has completed and the *account_info* field of the reply contains the final accounting information.

Communication between the SM, RRM, and the TPM

This section describes the communication that takes place between the TIE-In Profile Manager and the Session Manager and Resource Request Manager.

Add Request

The Session Manager sends add requests to the TIE-In Profile Manager for a variety of reasons as discussed in the "TIE-In Profile Manager Process Description" section. The *info* field in the reply contains additional information about the status of the request, such as why the request failed.

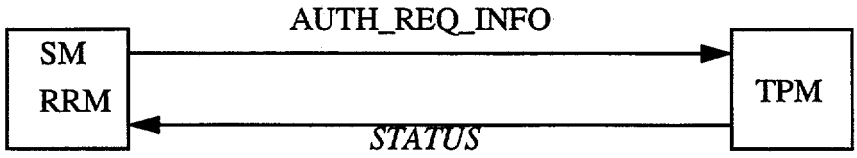


where *STATUS* is one of the following:

AUTH_REPLY_SUCCESS	The information was successfully added or updated.
AUTH_REPLY_NOENTITY	The specified user or application does not exist.
AUTH_REPLY_FAILED	The information could not be added or updated.

Information Request

Both the Session Manager and the Resource Request Manager obtain a variety of information from the database by sending AUTH_REQ_INFO type messages to the TPM, as discussed in the previous section. The *info* field in the reply contains additional information about the status of the request, such as why the request failed.

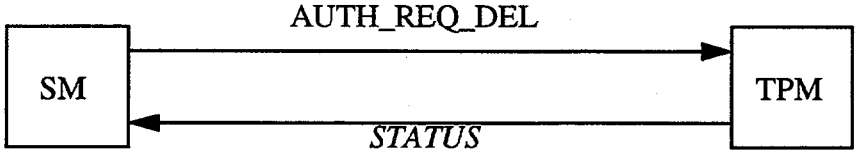


where *STATUS* is one of the following:

AUTH_REPLY_SUCCESS	The information was successfully obtained.
AUTH_REPLY_NOENTITY	The specified user or application does not exist.
AUTH_REPLY_FAILED	The information could not be obtained.

Delete Request

The Session Manager sends delete requests to the TPM in order to delete a user, an application, or a user's registration. The *info* field in the reply contains additional information about the status of the request, such as why the request failed.



where *STATUS* is one of the following:

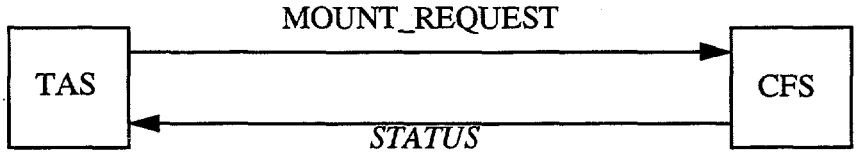
AUTH_REPLY_SUCCESS	The user, application, or registration was successfully deleted.
AUTH_REPLY_NOENTITY	The specified user, application, or registration does not exist.
AUTH_REPLY_FAILED	The user, application, or registration could not be deleted.

Communication Between the CFS and the TAS

This section describes the communication between the Common File System and the TIE-In Application Server. The TAS will send messages to the *mnt_auth* process of the CFS, and the *mnt_auth* process returns a reply to the message. The TAS sends two requests: mount and unmount.

Mount Request

Prior to executing the mount command on the application server, the TAS will send a message to *mnt_auth* to inform it of the impending mount. The *mnt_auth* process will verify some of the information in the message, extract some of the information, and send a reply back to the TAS.



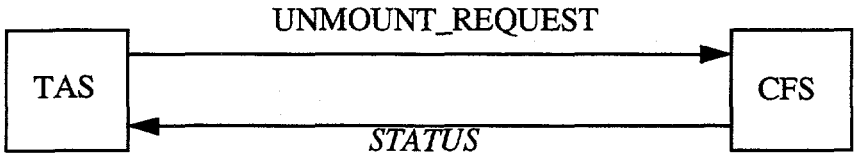
where *STATUS* is one of the following:

MNT_AUTH_REPLY_NOSUCH_DIR	The specified proxy directory does not exist.
MNT_AUTH_REPLY_PERM	The specified user or client does not have permission to mount the specified proxy directory.
MNT_AUTH_REPLY_TIME_ERR	The request from the TAS has exceeded the time limit set for the duration between the initial message sent by the SM to mnt_auth and the mount request message.
MNT_AUTH_REPLY_SUCCESS	The request was successfully processed.

Unmount Request

Prior to executing the umount command on the application server, the TAS will send a message to *mnt_auth* to inform it of the impending unmount. The *mnt_auth* process will

verify some of the information in the message, extract some of the information, and send a reply back to the TAS.

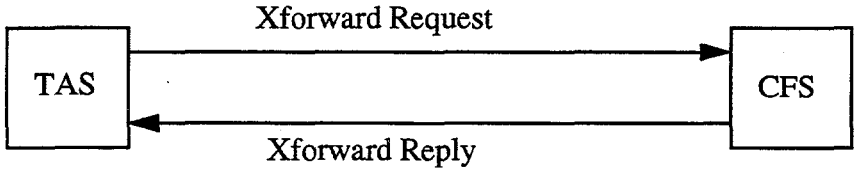


where *STATUS* is one of the following:

MNT_AUTH_REPLY_NOSUCH_DIR	The specified proxy directory does not exist.
MNT_AUTH_REPLY_PERM	The specified user or client does not have permission to unmount the specified proxy directory.
MNT_AUTH_REPLY_SUCCESS	The request was successfully processed.

Communication Between the SM and Xforward

This section describes the communication between the Session Manager and the Xforward process. The SM sends a variety of requests to the xforward process, which formulates and returns the requested information.



Inside the request is an operation. The possible operations and the value returned in the reply are shown in Table 23. The return value is actually a one-byte string indicating success or failure (XFORWARD_SUCCESS or XFORWARD_FAILED) followed by a string containing the specified information.

Table 23. Xforward request operation types and return values

Operation Type	Return value
XFORWARD_ADD	the proxy display
XFORWARD_ADD_SECURE	the proxy display and the magic cookie
XFORWARD_ADD_UDP	the proxy port
XFORWARD_ADD_TCP	the proxy port
XFORWARD_ADD_NO_NOTIFY	the proxy display
XFORWARD_ADD_SECURE_NO_NOTIFY	the proxy display and the magic cookie
XFORWARD_ADD_NFS_PROXY	the proxy home directory
XFORWARD_CLOSE	status (success/failure)
XFORWARD_DELETE	status (success/failure)

Intentionally Left Blank

Conclusion

TIE-In provides a new mechanism for connecting a distributed set of users with a distributed set of applications in an authenticated, secure manner. The TIE-In approach focuses on providing the non-expert with guided solutions embedded in intelligent user interfaces, while minimizing the investment required to utilize these technologies.

TIE-In has been in operation since June 1994. It currently has 29 integrated applications, of which 21 are available to both internal and external customers. One of the applications resides on a workstation at Lawrence Livermore National Laboratory, while the rest of the applications reside on Sandia workstations. There are a total of 650 registered TIE-In users, with approximately 45 users accessing the TIE-In system to successfully complete over 200 application requests per week.

Future Developments

In addition to continuing to bring a wide variety of applications into the TIE-In environment, there are a number of future developments that will help align TIE-In with the future of distributed computing.

DCE Applications

One probable future development effort of TIE-In is to make the various manager processes (Accounting Manager, Profile Manager, Request Manager, and Application Server) into DCE (Distributed Computing Environment) applications. This will allow the manager processes to be used as network resources. Each manager process will have an application programming interface (API) that will allow other projects to easily use TIE-In's existing processes.

In addition, the Distributed File System (DFS) component of DCE will provide a more secure alternative to TIE-In's current Common File System. This is because it provides a checksum with each file, which prevents the modification of a file during transmission between the server and the client.

This development effort is dependent upon having a DCE infrastructure, which is not in place at this time. The infrastructure is, however, under development not only at Sandia, but also at Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and the Defense Programs (DP) plants to support major DP projects like Advanced Design and Production Technologies (ADaPT) and the Accelerated Super Computing Initiative (ASCI).

CORBA Applications

Another possible future development effort of TIE-In is to make the various manager processes into CORBA (Common Object Request Broker Architecture) applications. This approach would also allow the manager processes to be used as network resources. It would be possible in the current environment to move towards CORBA applications. However, there is nothing in the CORBA standard that addresses security issues, such as using checksums, encryption, or Kerberos to secure the communication between the processes. Thus, by integrating CORBA distributed application efforts to work with TIE-In, these security requirements can be addressed.

Supporting Different Platforms

Currently, TIE-In provides Application Server processes for SunOS, Solaris, HP-UX, and SGI-IRIX systems. An investigation is currently underway to determine the feasibility of porting the TIE-In Application Server to the Windows NT platform. This would allow applications to run under Windows NT in addition to UNIX. This would be a valuable addition to TIE-In, as Windows NT applications are becoming more popular.

References

[1]S.P. Miller, B.C. Neuman, J.I. Schiller, and J.H. Saltzer, *Section E.2.1: Kerberos authentication and authorization system, Project Athena Technical Plan*, MIT Project Athena, Cambridge, Mass, Dec. 1987.

[2]Sun Microsystems, Inc., *Network Filesystem Specification*, RFC-1094, DDN Network Information Center, SRI International, Menlo Park, CA.

Intentionally Left Blank

Bibliography

Computer Security

William R. Cheswick and Steven W. Bellovin, *Firewalls and Internet Security*, Addison-Wesley, New York, 1994.

D. Borman, *Telnet Authentication: Kerberos Version 4*, RFC-1411, DDN Network Information Center, SRI International, Menlo Park, CA.

J. Kohl, B. Neuman, *The Kerberos Network Authentication Service (V5)*, RFC-1510, DDN Network Information Center, SRI International, Menlo Park, CA.

S.P. Miller, B.C. Neuman, J.I. Schiller, and J.H. Saltzer, *Section E.2.1: Kerberos authentication and authorization system, Project Athena Technical Plan*, MIT Project Athena, Cambridge, Mass, Dec. 1987.

Jeffrey I. Schiller, *Secure Distributed Computing*, Scientific American, November 1994, pp. 72 - 76.

Databases

Ingres Corporation, *INGRES Database Administrator's Guide*, Release 6.4, December 1991.

Ingres Corporation, *INGRES/Embedded SQL Companion Guide for C*, Release 6.4, December 1991.

Ingres Corporation, *INGRES/SQL Reference Manual,* Release 6.4, December 1991.

Patrick O'Neil, *Database Principles, Programming, and Performance*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1994.

Distributed Computing Environment

Harold W. Lockhart, Jr., *OSF DCE*, McGraw-Hill, Inc., New York, 1994.

Ward Rosenberry, David Kenney, and Gerry Fisher, *Understanding DCE*, O'Reilly and Associates, Inc., Sebastapol, CA, 1992.

John Shirley, Wei Hu, and David Magid, *Guide to Writing DCE Applications*, O'Reilly and Associates, Inc., Sebastapol, CA, 1994.

Network File System

John Bloomer, *Power Programming with RPC*, O'Reilly and Associates, Inc., Sebastapol, CA, 1992.

Hal Stern, *Managing NFS and NIS*, O'Reilly and Associates, Inc., Sebastapol, CA, 1991.

Hewlett-Packard, Co., *Programming and Protocols for NFS Services*, Fort Collins, CO, 1989.

Hewlett-Packard, Co., *Using NFS Services*, Fort Collins, CO, 1989.

Sun Microsystems, Inc., *Network Filesystem Specification*, RFC-1094, DDN Network Information Center, SRI International, Menlo Park, CA.

Programming

David A. Curry, *Using C on the UNIX System*, O'Reilly and Associates, Inc., Sebastapol, CA, 1991.

X Window System

Paul J. Asente and Ralph R. Swick, *X Window System Toolkit,* Digital Press, 1990.

David Flanagan, *X Toolkit Intrinsics Reference Manual*, O'Reilly and Associates, Inc., Sebastapol, CA, 1992.

Dan Heller and Paula Ferguson, *Motif Programming Manual*, O'Reilly and Associates, Inc., Sebastapol, CA, 1994.

Paula Ferguson and David Brennan, *Motif Reference Manual,* O'Reilly and Associates, Inc., Sebastapol, CA, 1993.

Adrian Nye, *Xlib Programming Manual*, O'Reilly and Associates, Inc., Sebastapol, CA, 1992.

Adrian Nye, *Xlib Reference Manual*, O'Reilly and Associates, Inc., Sebastapol, CA, 1993.

Adrian Nye and Tim O'Reilly, *X Toolkit Intrinsics Programming Manual, Motif Edition*, O'Reilly and Associates, Inc., Sebastapol, CA, 1992.

Robert Scheifler and Jame Gettys, *X Window System,* Digital Press, 1992.

Appendix A: Message Formats

Data Type Definitions

U32INT:		Unsigned 32 bit integer
8INT:		Unsigned 8 bit integer
NBO:		Network Byte Order - Most significant byte transmitted first.
U32NBOINT:		Unsigned 32 bit Network Byte Order integer
STRING:		Sequential array of bytes.
DATA:		
	length	U32INT
	data	STRING (of size length)
NBODATA:		
	length	U32NBOINT
	data	STRING (of size length)
ACCTDATA:		
	type	8INT 1: Storage 2: Memory 3: CPU 4: Connect 5: TA definable 6: TA definable 7: TA definable
	units	U32INT
	unit_cost	NBODATA
NBOACCTDATA:		
	type	8INT 1: Storage 2: Memory 3: CPU 4: Connect 5: TA definable 6: TA definable 7: TA definable

	units	NBOU32INT
	unit_descr	NBODATA

char_string: Sequential array of bytes terminated by zero valued byte.

USER_PROFILE

full_name	NBODATA	The user's full name.
comp_id_num	NBODATA	The user's company id number.
---	---	---
The following fields are used when the user is a Sandia employee.		
---	---	---
org_num	NBODATA	The user's organization number.
org_name	NBODATA	The user's organization name.
emp_num	NBODATA	The user's employee number.
mail_stop	NBODATA	The user's mail stop.
---	---	---
The following fields are used when the user is not a Sandia employee.		
---	---	---
department	NBODATA	The user's department.
street	NBODATA	The user's street address.
po_box	NBODATA	The user's post office box.
city	NBODATA	The user's city.
state	NBODATA	The user's state.
zip_code	NBODATA	The user's zip code.
country	NBODATA	The user's country.
---	---	---
email_addr	NBODATA	The user's electronic mail address.
phone	NBODATA	The user's phone number.
fax	NBODATA	The user's fax number.
citizenship	NBODATA	The user's country of citizenship.
visa_num	NBODATA	If the user is not a US citizen, they must provide a visa or passport number.
ssn	NBODATA	The user's social security number if he is a US citizen.
home_dir	NBODATA	The user's home directory that may be mounted by an application.
realm	NBODATA	The user's kerberos realm.
df_type	U32INT	The user's distributed file type.
df_service_class	U32INT	The user's distributed file service class.
acct_limit	NBODATA	The user's expenditure limit.
service_class	U32INT	The user's service class.
acct_expr_date	NBODATA	The user's account expiration date.

CHARGE_ACCT		
acct_type	8INT	The account type, currently only TIE_ACCT_TYPE_CASE. Future values may include purchase orders, charge cards, bank accounts, etc.
acct_number	NBODATA	The account number.
CHARGE_ACCTS		
count	U32INT	The number of charge accounts.
contents	CHARGE_ACCT	The charge accounts.
KEY		
encsize	8INT	The value of encsize is the length of the original key. This is done because the encryption/decryption will change the length of the information. If the value is 0 then contents are not encrypted. Limitation: the size of the key must be <= 255 bytes.
contents	NBODATA	The key in NBO.
KEYBLOCK		
key	KEY	
type	8INT	
CREDENTIALS		
type	8INT	Type of user credentials, one of TIE_AUTH_NONE: 0 TIE_AUTH_KRB4: 1 TIE_AUTH_KRB5: 2
contents	NBODATA	The credentials.
endtime	U32INT	The time at which the credentials are not longer valid.
sessionkey	KEYBLOCK	The session key.
tkt_cred	VOID *	Ticket pointer.
ENC_CREDS		Encrypted CREDENTIALS.
CKSUM		
type	8INT	Type of checksum. Acceptable types are TIE_CKSUM_NONE 0

		TIE_CKSUM_CRC_32 1 TIE_CKSUM_RSA_MD4 2 TIE_CKSUM_RSA_MD5 3
encsize	8INT	The value of encsize is the length of the original checksum. This is done because the encryption/decryption will change the length of the information. Limitation - the size of the key must be <=255
contents	NBODATA	Checksum of this request excluding the cksum label. A Cksum type of NONE will have a contents.length of zero.
CONSULTANT		
user_name	NBODATA	The user name of the consultant.
full_name	NBODATA	The full name of the consultant.
email_addr	NBODATA	The email address of the consultant.
phone_num	NBODATA	The phone number of the consultant.
CONSULTANTS		
consultants	CONSULTANT	The list of consultants.

TIE_RRM_REQ

Network IP/UDP send to port 1955

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_RRM_VERSION.
type	8INT	Type of request. Acceptable values; RRM_SUBMIT_REQ 1 RRM_STATUS_REQ 2 RRM_DELETE_REQ 3 RRM_ADD_ACCT 4 RRM_DEL_ACCT 5
username	NBODATA	Name of user requesting application. Must be unique, and assigned by the Network Database. Lower case only.
userid	U32NBOINT	User Id of username. Must be unique, assigned by the Network Database.
gids		
count	U32NBOINT	Number of Group ids to follow.
gid1	U32NBOINT	Group Id 1 (Registered in the 13900 Database)
.		
.		
gidn	U32NBOINT	Group Id n where n = count
resourcename	NBODATA	Name of application requested. Lower case only.
charge_acct	CHARGE_ACCT	The account to charge for the cost of the task.
resource_address		
type	8INT	Type of address contained in resource_addresses.address.data. Acceptable values; TIE_ADDR_TYPE_NONE 0 TIE_ADDR_TYPE_IP 1
address	NBODATA	The address in ascii representation. Example: the IP address 132.175.133.1 will have a length of 14, followed by the above null terminated string.
auth_levels	U32NBOINT	Authentication levels supported by this remote system. Acceptable values can be established by performing a logical OR of the following values,

		TIE_AUTH_NONE: 0 TIE_AUTH_KRB4: 1 TIE_AUTH_KRB5: 2 TIE_AUTH_SID: 4
credentials	ENC_CREDS	The user's credentials.
rrm_credentials	CREDENTIALS	The RRM's credentials.
session_logger	NBODATA	Port through which messages are logged
user_address		Users local account address. By default it is the address from where the connection originated. This may not be the same as the display.
type	8INT	Type of address contained in user_address.address.data. Acceptable values; TIE_ADDR_TYPE_IP 1
address	NBODATA	The address in NBO. Example: an IP address will have a length of 4, followed by the 32 bit NBO value of the address.
auth_levels	U32NBOINT	Ignored by RRM
req_id	U32NBOINT	Value assigned by the RRM for this task. For a request type of SUBMIT this value is zero.
tirrs_req_id	U32NBOINT	Value assigned by the TAS, which identifies this task from the TAS point of view. For a request type of SUBMIT this value is zero.
request_flags	U32NBOINT	Flags associated with this request.
xserver_key	KEY	The magic cookie key presented by X application clients to the proxy X server.
xfer_key	KEY	(Future use) Data surety key.
bandwidth	U32NBOINT	Maximum bandwidth of user's network connection. Used for applications that may have bandwidth requirements. A value of zero means unknown.
display	NBODATA	User's display name.
user_profile	USER_PROFILE	Info about the user.
forwarded_creds	ENC_CREDS	Credentials forwarded from the realm in which the user initially performed authentication.
extensions	NBODATA	For future extensions to the protocol.

cksum	CKSUM	The checksum for this request, which will always be encrypted.

TIE_RRM_REPLY

Network IP/TCP send to a defined port > 1023

Label	Value	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_RRM_REPLY.
type	8INT	Reply for request type of;
		RRM_SUBMIT_REQ 1
		RRM_STATUS_REQ 2
		RRM_DELETE_REQ 3
		RRM_ADD_ACCT 4
		RRM_DEL_ACCT 5
status	8INT	Return status of request. Values:
		RRM_SUCCESS 1
		RRM_AUTHFAIL 2
		RRM_XFAIL 3
		RRM_REQUNK 4
		RRM_REQCOMPL 5
		RRM_RESRC_UNAVAIL 6
		RRM_REQ_EXISTS 7
		RRM_REQ_UPDATE 8
		RRM_RESRC_UNK 9
info	NBODATA	Description from TAS providing additional information with regard to status.
username	NBODATA	Name of user requesting application. Must be unique and assigned by the Network Database. Lower case only.
userid	U32NBOINT	User Id of username. Must be unique and assigned by the Network Database.
resourcename	NBODATA	Name of application requested. Lower case only.
req_id	U32NBOINT	Value assigned by the RRM for this task. For a request type of SUBMIT this value is zero.
tirrs_req_id	U32NBOINT	Value assigned by the TAS, which identifies this task from the TAS point of view. For a request type of SUBMIT this value is zero.
extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The checksum for this reply, which will always be encrypted.

TIE_TIRRS_REQ

Network IP/UDP send to port 1955

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_TIRRS_REQ.
type	8INT	Type of request. Acceptable values; TIRRS_SUBMIT_REQ 1 TIRRS_STATUS_REQ 2 TIRRS_DELETE_REQ 3 TIRRS_ADD_ACCT 4 TIRRS_DEL_ACCT 5 TIRRS_COMPL_ACK 32
username	NBODATA	Name of user requesting application. Must be unique and assigned by the Network Database. Lower case only.
userid	U32NBOINT	User Id of username. Must be unique and assigned by the Network Database.
gids*		
count	U32NBOINT	Number of Group ids to follow.
gid1	U32NBOINT	Group Id 1 (Registered in the Network Database)
.		
.		
gidn	U32NBOINT	Group Id n where n = count
resourcename	NBODATA	Name of application requested. Lower case only.
user_address*		Users local account address. By default it is the address from where the connection originated. This may not be the same as the display.
type	8INT	Type of address contained in user_address.address.data. Acceptable values: TIE_ADDR_TYPE_IP 1
address	NBODATA	The address in ascii representation. Example: the IP address 132.175.133.1 will have a length of 14, followed by the above null terminated string.
auth_levels	U32NBOINT	Ignored
credentials	ENC_CREDS	The user's credentials.

req_id	U32NBOINT	Value assigned by the RRM for this task.
tirrs_req_id	U32NBOINT	Value assigned by the TAS, which identifies this task from the TAS point of view. For a request type of SUBMIT this value is zero. Request types of STATUS_REQ and DELETE_REQ should have non-zero values.
request_flags	U32NBOINT	Flags associated with this request.
xserver_key**	KEY	The key presented by X application clients to the user's X server.
xfer_key**	KEY	(Future use) Data surety key.
bandwidth*	U32NBOINT	Maximum bandwidth of user's network connection. Used for applications that may have bandwidth requirements. A value of zero means unknown.
display*	NBODATA	User's display name.
user_profile	USER_PROFILE	Info about the user.
forwarded_creds	ENC_CREDS	Credentials forwarded from the realm in which the user initially performed authentication.
extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The checksum of the request. This will always exist and be encrypted.

* These fields are ignored on status and acknowledgment type of requests.

** These fields may have a contents.length of zero, which indicates these keys are not applicable.

TIE_TIRRS_REPLY

Network IP/UDP send to port 1956

Label	Value	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_TIRRS_REPLY.
type	8INT	Reply for request type of TIRRS_SUBMIT_REQ 1 TIRRS_STATUS_REQ 2 TIRRS_DELETE_REQ 3 TIRRS_ADD_ACCT 4 TIRRS_DEL_ACCT 5 This information should have been obtained from the TIE_TIRRS_REQ message.
status	8INT	Return status of request. Acceptable values TIRRS_RRM_SUCCESS 1 TIRRS_RRM_AUTHFAI: 2 TIRRS_RRM_XFAIL 3 TIRRS_RRM_REQUNK 4 TIRRS_RRM_REQCOMPL 5 TIRRS_RRM_RESRC_UNAVAIL 6 TIRRS_RRM_REQ_EXISTS 7 TIRRS_RRM_REQ_UPDATE 8 TIRRS_RRM_RESRC_UNK 9
info	NBODATA	Description from TAS providing additional information with regards to status.
username	NBODATA	Name of user requesting application. This information should have been obtained from the TIE_TIRRS_REQ message.
userid	U32NBOINT	User Id of username. This information should have been obtained from the TIE_TIRRS_REQ message.
resourcename	NBODATA	Name of application requested. This information should have been obtained from the TIE_TIRRS_REQ message.
timestamp	NBODATA	Timestamp to identify when the reply was sent.
req_id	U32NBOINT	Value assigned by the RRM for this

		task. This value should be obtained from the TIE_TIRRS_REQ message.
tirrs_req_id	U32NBOINT	Value assigned by the TAS in response to a submit request. This value identifies this task from the TAS point of view.
credentials	CREDENTIALS	The credentials of the TAS.
account_info		
count	U32NBOINT	Number of entries to follow.
entry1	NBOACCTDATA	1st accounting information.
.		
.		
.		
entryn	NBOACCTDATA	Nth accounting record, where n = count.
extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The checksum for this request, which will always be encrypted.

TIE_AUTH_REQ

Network IP/TCP send to port 1956

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_AUTH_REQ.
type	8INT	Type of request. Acceptable values: AUTH_REQ_INFO 1 AUTH_REQ_ADD 2 (Updates if account exists) AUTH_REQ_DEL 3
name_type	8INT	The type of name presented (all actual flag names are preceded with TIE_AUTH_NAMETYPE_): USER 1 RESOURCE 2 MF_USER 3 NMF_USER 4 MF_RESRC 5 ADM_RESRC 6 REG_USER 7
session_id	NBODATA	Unique identifier of the user's TIE-In session
name	NBODATA	Name of user requesting application. Must be unique and assigned by the Network Database. Lower case only.
match_flags	U32NBOINT	Flags used to find matching users or applications.
credentials		The credentials with a type of TIE_AUTH_KRB5.
The following records are only necessary for Add requests with name type of TIE_AUTH_NAMETYPE_USER.		
auth_levels	8INT	Authentication levels supported. Acceptable values can be established by performing a logical OR of the following values TIE_AUTH_KRB5: 2 TIE_AUTH_SID: 4
userid	U32NBOINT	The user's User Id. Must be unique and assigned by the Network Database.
flags	U32NBOINT	Flags defining the state of this

		account TIE_USER_FLAGS_ACTIVE 1 TIE_USER_FLAGS_EXPIRED 2 TIE_USER_FLAGS_PW_- EXPIRED 4
gids		
count	U32NBOINT	Number of Group ids to follow.
gid1	U32NBOINT	Group Id 1 (Registered in the Network Database)
.		
.		
gidn	U32NBOINT	Group Id n where n = count
user_profile	USER_PROFILE	Information about the user.
def_charge_acct	CHARGE_ACCT	The user's default charge account.
charge_accts	CHARGE_ACCTS	All of the user's charge accounts.
user_resources		List of applications this user has authorized access to
count	8INT	The number of resource.name records to follow. The record name is repeated "count" times.
name	NBODATA	The name of the application.
description	NBODATA	(Ignored by AuthM)
acct_type	8INT	Default Account number type to use against this application.
acct_number	NBODATA	Default account number to use against this application.

The following records are only necessary for Add requests for name types of TIE_AUTH_NAMETYPE_RESOURCE, TIE_AUTH_NAMETYPE_APP_ACCESS, and TIE_AUTH_NAMETYPE_APP_CONS.

---

resource		
name	NBODATA	The name of the application.
description	NBODATA	Description of the application.
groups	NBODATA	The groups to which this application belongs.
flags	U32NBOINT	Flags defining the state of this application TIE_RSRC_FLAGS_ACTIVE 1 TIE_RESRC_FLAGS_REQ_DEL 2

			TIE_RESRC_FLAGS_KEY_-EXPIRED 4
	administrator	NBODATA	User name of application administrator
	def_account_info		
	count	U32NBOINT	Number of records to follow
	type	8INT	Accounting record type
	unit_descr	NBODATA	Description of type
	charge	NBODATA	Advertised charge per unit
	credit_acct_info	CHARGE_ACCT	The account to be credited when payment is received from the user's of the application.
resource_addresses			Lists the addresses of systems supporting this application. This record is only examined for Add requests with a name_type of RESOURCE.
	count	8INT	The number of address records to follow. The records type, contents, and auth_type are repeated "count" times.
	type	8INT	Type of address contained in resource_addresses[n].address.data. Acceptable values: TIE_ADDR_TYPE_IP 1
	address	NBODATA	The address in ascii representation. Example: the IP address 132.175.133.1 will have a length of 14, followed by the above null terminated string.
	service_class	U32NBOINT	The type of service offered by this application server.
	auth_levels	8INT	Type of authentication this entity supports. See above for possible values.
	realm	NBODATA	The realm in which this application server resides.
consultants		CONSULTANTS	The list of users who are consultants for this application.

The following records are only necessary for Add requests for name type of TIE_AUTH_NAMETYPE_REG_USER.

---

flags	U32NBOINT	A combination of the following flags that define the user. All flags are prefixed with TIE_AUTH_USER_FLAG: SANDIA_EMP, US_CITIZEN, SID_REQUESTED, BY_FAX, BY_MAIL, ARCHIVED
usernames	NBODATA	A listing of up to three of the user's preferred login names.
passwd	NBODATA	The user's initial password.
charge_accts	CHARGE_ACCTS	The user's charge accounts.
user_profile	USER_PROFILE	The user's profile information.

The following records are only necessary for Add requests for name type of TIE_AUTH_NAMETYPE_PROJECT.

---

project_id	NBODATA	The identification string of the project to be added.
description	NBODATA	A description of the project.
manager	NBODATA	The username of the project manager.
status	U32NBOINT	The status of the project: used to indicate whether or not the project is active.
charge_acct	CHARGE_ACCT	The charge account to which charges incurred by the project will be charged.
access_list		List of the usernames of the project members.
count	8INT	The number of user name records to follow.
name	NBODATA	The username of the user.

The following records are only necessary for Add requests for name type of TIE_AUTH_NAMETYPE_SESSION.

---

session_id	NBODATA	The identification string of the session to be added.
display	NBODATA	The display of the user who initiated the session.
start_time	NBODATA	The time the session was initiated.
fes	NBODATA	The Front End System that the user

		logged into to initiate the session.
user_name	NBODATA	The username of the user that initiated the session.
extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The cksum for this request. This will always be encrypted.

TIE_AUTH_REPLY

Network IP/TCP

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_AUTH_REPLY.
status	8INT	Results of request:
		AUTH_REPLY_SUCCESS 1
		AUTH_REPLY_NOENTITY 2
		AUTH_REPLY_FAILED 3
name_type	8INT	The type of name presented (all actual flag names are preceded with TIE_AUTH_NAMETYPE_):
		USER 1
		RESOURCE 2
		USER 3
		NMF_USER 4
		MF_RESRC 5
		ADM_RESRC 6
		REG_USER 7
name	NBODATA	Name of user or application.
info	NBODATA	Information provided by Authorization Manager
credentials		Credentials
type	8INT	Type of credentials, one of
		TIE_AUTH_KRB5: 2
contents	NBODATA	Network encoded credentials.

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_USER

---

auth_levels	8INT	Authentication levels supported by this remote system. Acceptable values can be established by performing a logical OR of the following values,
		TIE_AUTH_NONE: 0
		TIE_AUTH_KRB4: 1
		TIE_AUTH_KRB5: 2
		TIE_AUTH_SID: 4
userid	U32NBOINT	User Id of the user.
gids		
count	U32NBOINT	Number of Group ids to follow.

gid1	U32NBOINT	Group Id 1 (Registered int the Network Database)
.		
.		
gidn	U32NBOINT	Group Id n where n = count
charge_accts	CHARGE_ACCTS	The user's charge accounts.
user_resources		List of applications this user has authorized access to
count	8INT	The number of resource.name records to follow. The record name is repeated "count" times.
name	NBODATA	The name of the application.
description	NBODATA	Description of the application.
charge_acct	CHARGE_ACCT	The account number to charge for this application.
flags	U32NBOINT	Flags defining the state of this account. All flag names are prefixed with TIE_USER_FLAG_ and the values are in Hexadecimal.
		ACTIVE 1
		EXPIRED 2
		PW_EXPIRED 4
		SANDIA_EMP 8
		US_CITIZEN 10
		SID_REQUESTED 20
		BY_FAX 40
		BY_MAIL 80
		ARCHIVED 100
		ADMINSTRATOR 8000

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_RESOURCE

---

resource		
name	NBODATA	The name of the application.
description	NBODATA	Description of the application.
flags	U32NBOINT	Flags defining the state of this application
		TIE_RESRC_FLAGS_ACTIVE 1
		TIE_RESRC_FLAGS_REQ_DEL 2
		TIE_RESRC_FLAGS_KEY_-EXPIRED 4
administrator	NBODATA	User name of application administrator

def_account_info			
	count	U32NBOINT	Number of records to follow
	type	8INT	Accounting record type.
	unit_descr	NBODATA	Description of type.
	charge	NBODATA	Advertised charge per unit.
resource_addresses			Lists the addresses of systems supporting this application. This records is only examined for Add requests with a name_type of RESOURCE.
	count	8INT	The number of address records to follow. The records type, contents, and auth_type are repeated "count" times.
	type	8INT	Type of address contained in resource_addresses[n].address.data. Acceptable values: TIE_ADDR_TYPE_IP 1
	address	NBODATA	The address in ascii representation. Example: the IP address 132.175.133.1 will have a length of 14, followed by the above null terminated string.
	auth_levels	8INT	Type of authentication, this entity supports. See above for possible values.
consultants		CONSULTANTS	The list of users who are consultants for this application.

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_MF_USER or TIE_AUTH_NAMETYPE__NMF_USER

---

users			List of users matching or not matching the flags.
	count	8INT	The number of user name records to follow.
	name	NBODATA	The name of the user.
	flags	U32NBOINT	The flags defining the user.

The following information is returned on information requests and when the name_type is TIE_AUTH_NAMETYPE_MF_RESRC or

TIE_AUTH_NAMETYPE_ADM_RESRC

resources			List of applications that match the flags (in the MF_RESRC case) or the list of applications for which a specified user is the administrator (in the ADM_RESRC case).
	count	8INT	The number of resource.name records to follow. The record name is repeated "count" times.
	name	NBODATA	The name of the application.
	description	NBODATA	Description of the application.
	flags	U32NBOINT	Flags defining the state of this application TIE_RESRC_FLAGS_ACTIVE 1 TIE_RESRC_FLAGS_REQ_DEL 2 TIE_RESRC_FLAGS_KEY_-EXPIRED 4
	administrator	NBODATA	User name of application administrator
	def_account_info		
		count	U32NBOINT Number of records to follow
		type	8INT Accounting record type
		unit_descr	NBODATA Description of type
		charge	NBODATA Advertised charge per unit

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_REG_USER

count	U32INT	The number of registrations to follow.
contents		There will be "count" of the following records returned.
flags	U32NBOINT	A combination of the following flags that define the user. All flags are prefixed with TIE_AUTH_USER_FLAG: SANDIA_EMP, US_CITIZEN, SID_REQUESTED, BY_FAX, BY_MAIL, ARCHIVED
usernames	NBODATA	A listing of up to three of the user's preferred login names.
passwd	NBODATA	The user's initial password.

charge_accts	CHARGE_ACCTS	The user's charge accounts.
user_profile	USER_PROFILE	The user's profile information.

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_PROJECT

---

project_id	NBODATA	The identification string of the project whose information is being returned.
description	NBODATA	The description of the project.
manager	NBODATA	The username of the project manager.
status	U32NBOINT	The status of the project: used to indicate whether or not the project is active.
charge_acct	CHARGE_ACCT	The charge account to which charges incurred by the project will be charged.
access_list		List of the usernames of the project members.
count	8INT	The number of user name records to follow.
name	NBODATA	The username of the user.

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_PROJECTS

---

count	U32INT	The number of projects to follow.
contents		There will be "count" of the project records shown above returned.

The following information is returned on information requests when the name_type is TIE_AUTH_NAMETYPE_SESSION

---

count	U32INT	The number of sessions to follow.
contents		There will be "count" of the following session records returned.
session_id	NBODATA	The identification string of the session to be added.
display	NBODATA	The display of the user who initiated the session.
start_time	NBODATA	The time the session was initiated.

fes	NBODATA	The Front End System that the user logged into to initiate the session.
user_name	NBODATA	The username of the user that initiated the session.

---

extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The checksum for this reply, which will always be encrypted.

TIE_ACCT_REQ

Network IP/TCP send to port 1957

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_ACCT_REQ.
type	8INT	Type of request. Acceptable values; ACCT_ADD_REQ: 1 ACCT_UPDATE_REQ: 2 ACCT_INFO_REQ: 3
username	NBODATA	Username of the user for INFO type requests. For ADD and UPDATE requests, this is the username of the user who owns the task.
resourcename	NBODATA	For ADD and UPDATE requests, this is the name of the application on which the task is running.
req_id	U32NBOINT	For ADD and UPDATE requests, this is the task id assigned by the RRM.
credentials	ENC_CREDS	Credentials presented for authentication.

The following information is applicable only to ADD and UPDATE type requests.

---

userid	U32NBOINT	User Id of user.
tirrs_req_id	U32NBOINT	Task identifier assigned by the TAS.
state	8INT	State of the task: ACCT_STATE_ACTIVE ACCT_STATE_COMPLETE ACCT_STATE_ARCHIVED ACCT_STATE_BILLED ACCT_STATE_PAID
info	NBODATA	Information regarding the task.
charge_acct	CHARGE_ACCT	Account to which charges for this task will be billed.
account_info		
count	U32NBOINT	Number of entries to follow.
entry1	NBOACCTDATA	1st accounting information.
.		
.		
.		
entryn	NBOACCTDATA	Nth accounting record, where n = count.

The following information is applicable only to ADD and UPDATE type requests.

---

request_type		U32INT	Indicates which table(s) is to be searched. Values are ACCT_ACTIVE_INFO ACCT_COMPLETED_INFO ACCT_ARCHIVED_INFO ACCT_ALL_INFO
count		U32INT	The number of field descriptions to follow.
fields			There will be "count" fields. The fields are used to perform a search to find the desired tasks.
	field_name	NBODATA	The database name of the field.
	low_value	NBODATA	The low value of the field, used only for a range.
	value	NBODATA	The value of the field in all cases except for a range.
	high_value	NBODATA	The high value of the field, used only for a range.
	op_type	U32INT	The type of operation to for this field in the search: TIE_DB_EQUAL TIE_DB_GT, TIE_DB_LT, TIE_DB_RANGE
	table_mask	U32INT	Indicates which tables contain the field: TIE_DB_ACTIVE_TABLE 1 TIE_DB_COMPLETED_TABLE 2 TIE_DB_ARCHIVED_TABLE 4 TIE_DB_ALL_TABLES 7

---

extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The checksum for this request, which will always be encrypted.

TIE_ACCT_REPLY

Network IP/TCP

Label	Type	Description
length	U32NBOINT	Length of information to follow.
version	8INT	Version of TIE_ACCT_REPLY.
type	8INT	Type of request which this is a reply to. Acceptable values; ACCT_ADD_REQ 1 ACCT_UPDATE_REQ: 2 ACCT_INFO_REQ: 3
status	8INT	Return status of request. Acceptable values are: ACCT_REPLY_SUCCESS: 1 ACCT_REPLY_NOENTITY:2 ACCT_REPLY_FAILED: 3
info	NBODATA	Informational message returned by Account Manager, giving more details when status is not ACCT_REPLY_SUCCESS
credentials	CREDENTIALS	Credentials presented for authentication.
The following information is only returned for INFO type of requests		
count	U32NBOINT	The number of records to follow.
username	NBODATA	Name of task owner.
userid	U32NBOINT	User Id of user.
resourcename	NBODATA	Application used for this task.
req_id	U32NBOINT	Value assigned by the RRM.
tirrs_req_id	U32NBOINT	Value assigned by the TAS.
date	U32NBOINT	Date of last update (GMT).
state	8INT	If bit 0 is set to 1 then task is still active.
session_id	NBODATA	Unique string that identified the session from which the user submitted the task.
start_time	U32INT	The start time of the task.
stop_time	U32INT	The stop time of the task.
cost	NBODATA	The accrued cost of the task.
info	NBODATA	Message about the task.
comp_id_num	NBODATA	The id number of the company for which the user works. This is used to

		send out bills.
charge_acct	CHARGE_ACCT	The account to which the charges for this task will be billed.
account_info		
count	U32NBOINT	Number of entries to follow.
entry1	NBOACCTDATA	1st accounting information.
.		
.		
.		
entryn	NBOACCTDATA	Nth accounting record, where n = count.
extensions	NBODATA	For future extensions to the protocol.
cksum	CKSUM	The cksum for this reply, which will always be encrypted.

TIE_MNT_REQUEST

Network IP/TCP

Label	Type	Description
type	U32INT	Either TIE_FS_TYPE_MNT or TIE_FS_TYPE_UMNT.
proxy_dir	NBODATA	The proxy directory.
from_username	NBODATA	The username of the owner of the application process on the application server:
to_username	NBODATA	The user's TIE-In username, which must also be the username of the user on the file server.
from_uid	U32INT	The user id of the owner of the application process on the application server.
to_uid	U32INT	The user's TIE-In user id, which must also be the user id of the user on the file server.
credentials	CREDENTIALS	Credentials presented for authentication.

TIE_MNT_REPLY

Network IP/TCP

Label	Type	Description
type	U32INT	Either TIE_FS_TYPE_MNT or TIE_FS_TYPE_UMNT.
status	U32INT	Indicates the success or failure of the request.
info	NBODATA	Contains information about the status of the request.

Gateway Service Request

Network IP/TCP

Label	Type	Description
auth_type	U8INT	Either AUTH_KRB5, AUTH_KRB4, or AUTH_NONE.
credentials	CREDENTIALS	Credentials used to authenticate the user making the request.
server	STRING	Contains the server to allow access to.
operation	U8INT	One of the following: XFORWARD_ADD, XFORWARD_ADD_SECURE, XFORWARD_ADD_UDP, XFORWARD_ADD_TCP, XFORWARD_ADD_NFS_PROXY, XFORWARD_ADD_NO_NOTIFY, XFORWARD_ADD_SECURE_NO_NOTIFY, XFORWARD_CLOSE, XFORWARD_DELETE, XFORWARD_ADD_ICE
num_clients	U8INT	The number of clients.
client_names	STRING_ARRAY	Names of the clients. There will be num_client client names.

Gateway Service Reply

Network IP/TCP

Label	Type	Description
status	U8INT	Status of the request, either XFORWARD_SUCCESS or XFORWARD_FAILED.
result	STRING	The result depends upon the operation type. It will either be a proxy display, proxy port, or NFS initialization string.

Intentionally Left Blank

Appendix B: TIE-In User and System Requirements

This Appendix outlines some basic requirements for the configuration and use of TIE-In.

Router Subsystem

The router subsystem in general must allow the following interconnections between systems within the internal subnetwork and all Front End Systems:

* Tcp ports (at least one of 23, 513, 512), and ports > 1023 from systems within the internal subnetwork to all Front End Systems.
* Tcp port 751 from all TIE-In Application Servers within the internal subnetwork to the Kerberos Authentication Service (currently being handled from the Master Front End System).
* Tcp ports > 1023 from the Kerberos Authentication Service (currently being handled from the Master Front End System) to all TIE-In Application Servers within the internal subnetwork.
* Udp port 1957 from all TIE-In Application Servers within the internal subnetwork to the Master Front End System.
* Udp port 1956 from the Master Front End System to all TIE-In Application Servers within the internal subnetwork.
* Udp port 88 from all TIE-In Application Servers within the internal subnetwork to the Kerberos Authentication Service (currently being handled from the Master Front End System).
* Udp ports >1023 from the Kerberos Authentication Service (currently being handled from the Master Front End System), to systems within the internal subnetwork.

Users

Users of TIE-In must have access to workstations or PCs which have:

* TCP/IP
* X Windows Server software
* Internet Access, or have a Modem and SLIP capabilities.

Users should connect to Front End Systems from host systems which provide:

* Telnet or some means of connecting to tcp ports 513 or 512.
* Ftp server.

User Names and UserId

* TIE-In assigns usernames and userids to registered users to insure that they are unique. Systems participating as application hosts must adhere to those assignments.

System Names

* For all Front End Systems *tiein.auth, tiein.acct, tiein.rrm* must be referenced to tiein.sandia.gov in the local domain service or the /etc/hosts file.
* The IP address associated with any TIE-In Application Server must resolve to the full domain name of the system hosting the TIE-In Application Server. For example, if the system call gethostname() returns tiein.sandia.gov, and the TIE-In Application Server, running on tiein.sandia.gov is registered with the IP address 132.175.133.1, then the command "nslookup 132.175.133.1" should return the name "tiein.sandia.gov".

Time Synchronization

* Kerberos requires loosely synchronized clocks (within 5 minutes) between all Front End Systems and TIE-In Application Server host systems. The application program *tdset* will be provided which allows a system to synchronize its system clock to any host providing the "time" service through inetd (tcp port 37). Front End Systems and TIE-In Application Server hosts may also use the network time protocol (ntp) to synchronize their system clocks. Ntp servers which tiein.sandia.gov uses to synchronize its system clock to are:

  esavax.esa.lanl.gov
  heechee.esa.lanl.gov
  eagle.tamu.edu
  tick.cs.unlv.edu

Appendix C: Utility Programs and Scripts

TIE-In has a group of utility programs and scripts that are used to help maintain the system. Some of these are set up as cron jobs, while others are manually executed as needed. This appendix describes these utility programs and scripts.

The following programs and scripts are located in the */tiein/bin* directory:

auto_register - determines if any users have submitted a request for an account via the form on the TIE-In homepage on the World Wide Web. If there are any account requests, this program generates a registration form on behalf of the user.

check_regs - determines if there are any user registrations that need to be processed; it is called by the *notify.administrator* script.

count_users - simply counts the number of registered TIE-In users.

create_scis_recs - creates SCIS records so that customers are billed for running applications that are set up to charge customers on a per-use basis.

email_addrs - generates a file containing the e-mail addresses of all registered TIE-In users.

foreigns - outputs a list of non-Sandia systems that connected to the TIE-In Front End System.

fs_controller - starts the *mnt_server*, *nfs_server*, and *mnt_auth* processes.

fs_kill - shuts down the *mnt_server*, *nfs_server*, and *mnt_auth* processes.

get_Xfails - outputs a list of display names that had some sort of X-related failure.

get_case_numbers.sh - this script retrieves that latest version of the valid_cases.dat file, which is used to validate all CASE numbers provided by TIE-In users. It is run as a cron job every weeknight.

notify_administrator - this script calls the check_regs program. If there are any registrations to be processed, it displays the names of the people who have submitted registrations in an X notification window and displays it on the administrators X-server. The file /tiein/config/tiein_administrator.dat contains the display information for the TIE-In administrator.

print_stats - generates detailed statistics about how many users have accessed which applications, and when they accessed them.

shm_dump - displays the contents of the shared memory segments used by the *mnt_server*, *nfs_server*, and *mnt_auth* processes.

show_slip_users - outputs the list of users who have logged into the TIE-In terminal server during the current week, along with the number of times they have logged in.

show_tie_logins - outputs the list of users who have logged into the TIE-In Front End System during the current week.

taccess - displays all of the telnet connections made to the TIE-In Front End System for the current day.

tlogins - displays the list of users who are currently logged into the TIE-In Front End System.

The following scripts are located in the */usr/local/bin* directory:

tiein_back.sh - performs backups of the tiein machine. This script is run as a cron job once a week.

tiein_back_logs.sh - performs backups of tiein log files. It copies the current version of the many log files to the /tiein/archive directory, then removes all the information from the log files. This script is run as a cron job once a week.

The following scripts are executed as cron jobs by the *ingres* user:

ing_archive - moves all tasks that are older than the specified number of days from the *completed_tasks* and *completed_units* tables to the *archived_tasks* and *archived_units* tables. This script is executed every night.

ing_backup - performs a backup of the database tables. This script runs three times a week.

ing_err_monitor - monitors the log file */usr/adm/tiein.log* for any messages containing TIE_DB_ERROR. It then writes these messages to the log file */usr/adm/db_error.log* and sends an email message to the TIE-In administrators to let them know that a database error has occurred. This script runs every 30 minutes.

ing_modify - modifies the database tables so that the unused space in the tables is freed. This script runs once a week.

ing_table_defs - generates table definitions for all of the database tables. These definitions can be used to regenerate the tables in case they are lost. This script is only run twice a month.

ing_verify - runs the verify_cases and verify_employees scripts. It is run nightly.

verify_cases - checks each CASE number provided by TIE-In users against the *valid_cases.dat* file. If a CASE number does not appear in the file, the CASE number is marked as invalid and can no longer be used. This script is called by *ing_verify*.

verify_employees - checks all TIE-In users that are Sandia employees or contractors against the *employees.dat* file. If an employee or contractor is not in the file, his account is disabled and a message is sent to the TIE-In administrators to let them know that the user's account can be deleted.

Intentionally Left Blank

Appendix D: The TIE-In Database Tables

This Appendix discusses the Ingres database used by TIE-In to maintain user, application, and accounting information. The database resides on the TIE-In Front End System. All access to the database is made via calls to the Database Library, which contains functions written in C utilizing Embedded SQL. The functions are discussed in Appendix B.

User Information Tables

Information about TIE-In users is stored in four tables:

(1) *user_info* - the main table, containing the user's full name, address, phone, etc.

(2) *user_group_ids* - contains the list of the user's group ids

(3) *user_applications* - contains the user's application list

(4) *user_charge_accounts* - contains the user's charge account list

Tables 24-27 show the information contained in each of the above database tables. Note that the *user_group_ids*, *user_applications*, and *user_charge_accounts* tables may contain more than one entry for each user. For example, if the user has four applications on her application list, the *user_applications* table will contain four entries for the user, each one identifying one of her applications.

The database tables are accessed via the Database Library functions *tie_db_del_user*, *tie_db_get_user*, *tie_db_mod_user*, and *tie_db_put_user*, all of which are discussed in Appendix B.

Table 24. Description of the Database Table user_info

Field Name	Ingres Datatype	Description
user_name	varchar(15)	the user's assigned user name, consistent with the Network Database
user_id	integer	the user's assigned user id, consistent with the Network Database
user_full_name	varchar(31)	the full name of the user

Table 24. Description of the Database Table user_info *(Continued)*

Field Name	Ingres Datatype	Description
company_id_num	varchar(31)	a unique identifier associated with the user's company
org_num	varchar(15)	the user's organization number if he is a Sandia employee
org_name	varchar(63)	the user's organization name if he is a Sandia employee
mail_stop	varchar(15)	the user's mail stop if he is a Sandia employee
emp_number	varchar(15)	the user's employee number if he is a Sandia employee
department	varchar(63)	the user's department if he is not a Sandia employee
street_address	varchar(63)	the user's company address if he is not a Sandia employee
po_box	varchar(31)	the user's PO Box if he is not a Sandia employee
city	varchar(31)	the user's city if he is not a Sandia employee
state	varchar(23)	the user's state if he is not a Sandia employee
zip_code	varchar(15)	the user's zip-code if he is not a Sandia employee
country	varchar(31)	the user's country if he is not a Sandia employee
email_address	varchar(63)	the user's e-mail address
phone	varchar(15)	the user's phone number
fax	varchar(15)	the user's fax number
citizenship	varchar(31)	the user's country of citizenship
visa_pass_num	varchar(63)	the user's visa/passport number for non-US citizens
ssn	varchar(15)	the user's Social Security Number for US citizens
home_dir	varchar(63)	the user's home directory that can be mounted by applications
df_type	integer	the user's distributed file system type, such as NFS or NFS_PROXY

Table 24. Description of the Database Table user_info *(Continued)*

Field Name	Ingres Datatype	Description
df_service_class	integer	the class of service offered by the user's file system, for future use
realm	varchar(63)	the user's realm, for future use
def_charge_type	varchar(15)	the user's default charge type
def_charge_num	varchar(31)	the user's default charge number
acct_limit	varchar(31)	maximum amount of charges the user is allowed to accrue
flags	varchar(15)	various flags describing the user
auth_level	varchar(15)	the user's authorization level: Kerberos 5 and/or SecureID (actual values of 2, 4, or 6)
service_class	varchar(15)	the user's class of service - currently unused
acct_expr_date	varchar(31)	the expiration date of the user's account

Table 25. Description of the Database Table user_group_ids

Field Name	Ingres Datatype	Description
user_id	integer	the user's assigned user id, consistent with the Network Database
group_id	integer	one of the user's group ids

Table 26. Description of the Database Table user_charge_accounts

Field Name	Ingres Datatype	Description
user_id	integer	the user's assigned user id, consistent with the Network Database
charge_type	integer	the charge type for the corresponding charge number: CASE, PO, Visa, etc.

Table 26. Description of the Database Table user_charge_accounts

Field Name	Ingres Datatype	Description
charge_number	varchar(31)	the charge number: CASE number, PO number, Visa number, etc.

Table 27. Description of the Database Table user_applications

Field Name	Ingres Datatype	Description
user_id	integer	the user's assigned user id, consistent with the Network Database
application_name	varchar(31)	the name of the application
charge_type	integer	the charge type for the corresponding charge number: CASE, PO, Visa, etc.
charge_number	varchar(31)	the charge number: CASE number, PO number, Visa number, etc. - if specified, this is the charge number that will be charged when the user runs the corresponding application

Application Information Tables

Information about the TIE-In applications is stored in four tables:

(1) *application_info* - general information about the application
(2) *application_groups* - the groups to which the application belongs
(3) *application_addresses* - the addresses of the application
(4) *application_costs* - the cost information for the application
(5) *application_consultants* - the consultants for the application
(6) *application_access* - the list of users allowed to access the application

Tables 28-32 show the information contained in each of the above database tables. Note that the *application_groups*, *application_addresses*, *application_costs* and *application_consultants* tables may contain more than one entry for each application. For example, if the application can be reached at three addresses, the *application_addresses* table will contain three entries for the application, each one identifying one of the addresses.

The database tables are accessed via the Database Library functions *tie_db_del_rsrc*, *tie_db_get_rsrc*, *tie_db_get_rsrc_list*, *tie_db_get_rsrc_match*, *tie_db_get_rsrc_desc*, *tie_db_mod_rsrc*, *tie_db_put_rsrc*, and *tie_db_put_app_cons*, all of which are discussed in Appendix B.

Table 28. Description of the Database Table application_info

Field Name	Ingres Datatype	Description
name	varchar(31)	the name of the application
description	varchar(95)	a description of the application
administrator	varchar(15)	user_name of the application administrator
number_of_servers	integer	the number of servers on which the application is running
flags	integer	flags describing the application
credit_number	varchar(31)	a charge number to be credited upon receipt of payment from users
credit_type	integer	the type of the charge number, typically CASE, but could also be PO, etc.
line_number	varchar(15)	used to generate SCIS records for billing purposes; corresponds to the Service Center Activity

Table 29. Description of the Database Table application_groups

Field Name	Ingres Datatype	Description
name	varchar(31)	the name of the application
application_group	varchar(31)	the name of a group to which the application belongs

Table 30. Description of the Database Table application_addresses

Field Name	Ingres Datatype	Description
name	varchar(31)	the name of the application
server_number	integer	number indicating the order in which the various servers are contacted
server_address	varchar(31)	the address of the server running the application
realm	varchar(63)	the authentication realm of the server
address_type	integer	a flag indicating the type of address, such as IP or DECNET. The server can also be disabled using this flag.
authorization_level	integer	the level at which users must be authenticated in order to use the application, typically Kerberos 5 or SecurId
service_class	integer	the class of service provided by this address

Table 31. Description of the Database Table application_costs

Field Name	Ingres Datatype	Description
name	varchar(31)	the name of the application
unit_type	integer	an integer value, 1- 7, that defines what the units represent: storage, CPU usage, connect time, etc. See the description of the TAS Status file for the list of unit types
unit_description	varchar(63)	description of what the units represent
unit_cost	float	cost per unit, where a unit is a second

Table 32. Description of the Database Table application_consultants

Field Name	Ingres Datatype	Description
application_name	varchar(31)	the name of the application
consultant	varchar(15)	the user name of a consultant
priority	integer	the order in which the consultant should appear on the consultant list

Table 33. Description of the Database Table application_access

Field Name	Ingres Datatype	Description
application_name	varchar(31)	the name of the application
server_address	varchar(31)	the address of a server for the application
user_name	varchar(15)	the user name of a user allowed to access the application running on the specified server

Accounting Information Tables

The TIE-In accounting information is stored in six tables:

(1) *active_tasks* - general information about active tasks
(2) *active_units* - the accounting information for active tasks
(3) *completed_tasks* - general information about completed tasks
(4) *completed_units* - the accounting information for completed tasks
(5) *archived_tasks* - general information about archived tasks
(6) *archived_units* - the accounting information for archived tasks

Active tasks are those that are currently running on a TIE-In Application Server. The accounting information about these tasks is periodically updated in the database. Completed tasks are those that have completed within the past 14 days. In order to prevent the *completed_tasks* and *completed_units* tables from becoming too large, tasks that have been completed for more than 14 days are archived. When a task is archived, the information about the task is moved from the *completed_tasks* and *completed_units* tables to the *archived_tasks* and *archived_units* tables. When the TIE-In billing procedure is set

up, the tasks will be left on the completed list until they have been billed and payment has been received.

Tables 34-39 show the information contained in each of the above database tables. Note that the *active_units*, *completed_units*, and *archived_units* tables may contain more than one entry for each application. For example, if an application charges for three unit types, the *active_units* table will contain three entries for the task, each one identifying one of the unit charges.

The database tables are accessed via the Database Library functions *tie_db_put_req*, *tie_db_get_req*, *tie_db_del_req*, *tie_db_mod_req*, *tie_db_move_req*, *tie_db_archive*, and *tie_db_archive_req*, all of which are discussed in Appendix B.

Table 34. Description of the Database Table active_tasks

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify a task
tas_task_id	integer	numeric id assigned by the TAS to identify a task - typically the process id of the task
user_name	varchar(15)	user name of the user to whom the task belongs
user_id	integer	user id of the user to whom the task belongs
app_name	varchar(31)	the name of the application running the task
session_id	varchar(127)	a character string that identifies the user's session from which the task was submitted
company_id_num	varchar(31)	a string identifying the user's company
start_time	integer	the time the task was submitted to the TAS, in seconds since January 1, 1970
last_update	integer	the time the accounting information for the task was last updated, in seconds since January 1, 1970
charge_type	integer	the type of the charge_number, such as CASE, PO, Visa, etc
charge_number	varchar(31)	the user's charge number

Table 34. Description of the Database Table active_tasks *(Continued)*

Field Name	Ingres Datatype	Description
scis_case_num	varchar(15)	the charge number to which the cost will be transferred; this is the CASE number that appears in the SCIS record
server_addr	varchar(79)	the address of the server that is running the task
cost_accrued	float	the cost accrued for the task as of the last update
info	varchar(95)	a character string message about the state of the task

Table 35. Description of the Database Table active_units

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify the task
unit_type	integer	an integer value, 1- 7, that defines what the units represent: storage, CPU usage, connect time, etc. See the description of the TAS Status file for the list of unit types
units	integer	the number of units accrued by the task
unit_description	varchar(63)	description of what the units represent
unit_cost	float	cost per unit, where a unit is a second
total_cost	float	the total cost accrued so far for this unit type

Table 36. Description of the Database Table completed_tasks

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify a task
tas_task_id	integer	numeric id assigned by the TAS to identify a task - typically the process id of the task
user_name	varchar(15)	user name of the user to whom the task belongs
user_id	integer	user id of the user to whom the task belongs
app_name	varchar(31)	the name of the application running the task
session_id	varchar(127)	a character string that identifies the user's session from which the task was submitted
start_time	integer	the time the task was submitted to the TAS, in seconds since January 1, 1970
stop_time	integer	the time the task completed, in seconds since January 1, 1970
duration	integer	the number of seconds the task was running
charge_type	integer	the type of the charge_number, such as CASE, PO, Visa, etc.
charge_num	varchar(31)	the charge number to which the cost of the task will be billed
company_id_num	varchar(31)	a string identifying the user's company
scis_case_num	varchar(15)	the charge number to which the cost will be transferred; this is the CASE number that appears in the SCIS record
server_addr	varchar(79)	the address of the server that is running the task
total_cost	float	the total cost of the task
info	varchar(95)	a character string message about the state of the task

Table 36. Description of the Database Table completed_tasks *(Continued)*

Field Name	Ingres Datatype	Description
billed_flag	varchar(7)	indicates whether or not an SCIS record has been generated for this task
paid_flag	varchar(7)	indicates whether of not the user has paid for the task; currently unused
billed_date	varchar(31)	the date the SCIS record was generated for the task
paid_date	varchar(31)	the date payment was received for the task; currently unused

Table 37. Description of the Database Table completed_units

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify the task
tas_task_id	integer	numeric id assigned by the TAS to identify the task
start_time	integer	the start time of the task
unit_type	integer	an integer value, 1- 7, that defines what the units represent: storage, CPU usage, connect time, etc. See the description of the TAS Status file for the list of unit types.
unit_description	varchar(63)	description of what the units represent
unit_cost	float	cost per unit, where a unit is a second
units	integer	the number of units accrued by the task
total_cost	float	the total cost for this unit type

Table 38. Description of the Database Table archived_tasks

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify a task
tas_task_id	integer	numeric id assigned by the TAS to identify a task - typically the process id of the task
user_name	varchar(15)	user name of the user to whom the task belongs
user_id	integer	user id of the user to whom the task belongs
app_name	varchar(31)	the name of the application running the task
session_id	varchar(127)	a character string that identifies the user's session from which the task was submitted
start_time	integer	the time the task was submitted to the TAS, in seconds since January 1, 1970
stop_time	integer	the time the task completed, in seconds since January 1, 1970
duration	integer	the number of seconds the task was running
charge_type	integer	the type of the charge_number, such as CASE, PO, Visa, etc.
charge_number	varchar(31)	the charge number to which the cost of the task will be billed
scis_case_num	varchar(15)	the charge number to which the cost will be transferred; this is the CASE number that appears in the SCIS record
server_addr	varchar(79)	the address of the server that is running the task
company_id_num	varchar(31)	a string identifying the user's company
total_cost	float	the total cost of the task
info	varchar(95)	a character string message about the state of the task

Table 38. Description of the Database Table archived_tasks *(Continued)*

Field Name	Ingres Datatype	Description
billed_flag	varchar(7)	indicates whether or not an SCIS record has been generated for the task
paid_flag	varchar(7)	indicates whether of not the user has paid for the task; currently unused
billed_date	varchar(31)	the date the SCIS record was generated for the task
paid_date	varchar(31)	the date payment was received for the task; currently unused
archived_date	integer	the date the task was archived, in seconds since January 1, 1970

Table 39. Description of the Database Table archived_units

Field Name	Ingres Datatype	Description
task_id	integer	numeric id assigned by the RRM to identify the task
tas_task_id	integer	numeric id assigned by the TAS to identify the task
start_time	integer	the start time of the task
unit_type	integer	an integer value, 1- 7, that defines what the units represent: storage, CPU usage, connect time, etc. See the description of the TAS Status file for the list of unit types
unit_description	varchar(63)	description of what the units represent
unit_cost	float	cost per unit, where a unit is a second
units	integer	the number of units accrued by the task
total_cost	float	the total cost for this unit type

User Registration Tables

When a user submits a registration for a TIE-In account, the registration information is stored in three tables:

(1) *registration_info* - general information about the user
(2) *registration_charge_info* - the charge accounts the user submitted
(3) *registration_logins* - the list of the user's desired user names

After a TIE-In administrator has registered the user, the user's entries in the above tables are deleted. Tables 40-42 describe the TIE-In user registration tables. Note that the *registration_charge_info* table and the *registration_logins* table can have more than one entry for each user. For example, the TIE-In registration form allows user's to input up to three choices for user names. If a user inputs three user names, the *registration_logins* table will have three entries for the user.

Table 40. Description of the Database Table registration_info

Field Name	Ingres Datatype	Description
name	varchar(31)	the full name of the user
company_id_num	varchar(31)	a unique identifier associated with the user's company
org_num	varchar(15)	the user's organization number if he is a Sandia employee
org_name	varchar(63)	the user's organization name if he is a Sandia employee
mail_stop	varchar(15)	the user's mail stop if he is a Sandia employee
emp_number	varchar(15)	the user's employee number if he is a Sandia employee
department	varchar(63)	the user's department if he is not a Sandia employee
street_address	varchar(63)	the user's company address if he is not a Sandia employee
po_box	varchar(31)	the user's PO Box if he is not a Sandia employee
city	varchar(31)	the user's city if he is not a Sandia employee
state	varchar(23)	the user's state if he is not a Sandia employee

Table 40. Description of the Database Table registration_info *(Continued)*

Field Name	Ingres Datatype	Description
zip_code	varchar(15)	the user's zip-code if he is not a Sandia employee
country	varchar(31)	the user's country if he is not a Sandia employee
email_address	varchar(63)	the user's email address
phone	varchar(15)	the user's phone number
fax	varchar(15)	the user's fax number
citizenship	varchar(31)	the user's country of citizenship
visa_pass_num	varchar(63)	the user's visa/passport number for non-US citizens
ssn	varchar(15)	the user's Social Security Number for US citizens
flags	varchar(15)	various flags describing the user
password	varchar(15)	the user's initial password

Table 41. Description of the Database Table registration_charge_info

Field Name	Ingres Datatype	Description
ssn	varchar(15)	the social security number of the user for US citizens, the visa/passport number for non-US citizens
charge_type	integer	the charge type for the corresponding charge number: CASE, PO, Visa, etc.
charge_number	varchar(31)	the charge number: CASE number, PO number, Visa number, etc.

Table 42. Description of the Database Table registration_logins

Field Name	Ingres Datatype	Description
ssn	varchar(15)	the social security number of the user for US citizens, the visa/passport number for non-US citizens
username	varchar(15)	the user's desired username

Application Group Table

An application can belong to one or more groups. The *application_group_list* table keeps track of the existing groups, storing the group name and a brief description as shown in Table 43.

Table 43. Description of the Database Table application_group_list

Field Name	Ingres Datatype	Description
group_name	varchar(31)	the name of the group
description	varchar(63)	a description of the group

Charge Account Info Table

The information about valid charge accounts is stored in the *charge_account_info* table, which is shown in Table 44.

Table 44. Description of the Database Table charge_account_info

Field Name	Ingres Datatype	Description
charge_num	varchar(31)	the actual charge number
charge_type	integer	the type of charge number, typically either a CASE number, Service Order, or Project ID.

Table 44. Description of the Database Table charge_account_info

Field Name	Ingres Datatype	Description
project_id	varchar(31)	the project id with which this charge number is associated
status	integer	the status of the charge number - valid or invalid, etc.
load_factor	float	the load that is to be applied to this charge number when the user is billed; this field is not currently used
balance	money	the balance amount for this charge number

The Project Tables

The information about projects is stored in two tables:

(1) *project_info* - contains general information about the projects
(2) *project_access* - contains an access list for a project

These tables are described in Tables 45 and 46, respectively.

Table 45. Description of the Database Table project_info

Field Name	Ingres Datatype	Description
project_id	varchar(31)	the name or id of the project
description	varchar(95)	a description of the project
charge_num	varchar(31)	the charge number to which costs accrued by users in this project will be charged
charge_type	integer	the charge type of the charge number
manager	float	the username of the user who is the manager of the project
status	money	the status of the project

Table 46. Description of the Database Table project_access

Field Name	Ingres Datatype	Description
project_id	varchar(31)	the name or id of the project
user_name	varchar(15)	the username of a user who is part of the project

The Current Sessions Table

Information about which users have a current TIE-In session is stored in the current_sessions table, which is described in Table 47.

Table 47. Description of the Database Table current_sessions

Field Name	Ingres Datatype	Description
session_id	varchar(95)	the session id of this session
user_name	varchar(15)	the username of the user who owns this session
user_display	varchar(31)	the user's display
fes	varchar(63)	the TIE-In Front End System through which the user initiated the session
start_time	date	the time at which the session was initiated

DISTRIBUTION

20	MS	1110	Jim Ang, 9204
5		0807	Glenn Machin, 4918
20		0807	Rich Detry, 4918
1		0809	Paul Brooks, 4421
1		0439	Ed Marek, 9234
1		0807	Mike Cahoon, 4918
1		1109	Rich Pryor, 9202
1		0321	Bill Camp, 9200
1		0318	Milt Clauser, 9201
1		1111	Sudip Dosanjh, 9221
1		1109	Art Hale, 9224
1		1110	Dick Allen, 9222
1		1110	David Greenberg, 9223
1		1111	Grant Heffelfinger, 9226
1		0819	Mike McGlaun, 9231
1		0820	Paul Yarrington, 9232
1		0841	Paul Hommert, 9100
1		0833	Johnny Biffle, 9103
1		0443	Hal Morgan, 9117
1		0836	Carl Peterson, 9116
1		0458	Bob Thomas, 5100
1		0322	Pat Eicker, 9600
1		0949	Ray Harrigan, 9602
1		1176	Rob Palmquiest, 9651
1		0951	Dave Strip, 9621
1		0660	Margaret Olson, 9622
1		0507	Kathleen McCaughey, 9700
1		0431	Sam Varnado, 9400
1		0163	Joe Polito, 9800
1		9003	Dona Crawford, 8900
1		9011	Rich Palmer, 8901
1		9011	Jim Costa, 8920
1		9011	Peter Dean, 8910
1		0811	Doug Brown, 4621
1		0458	Jim Asay, 5132
1		0458	Laura Gilliom, 5133
1		0472	A. Kay Hays, 5136
1		0458	J. Steve Rottler, 9003
1		1427	Peter Mattern, 1100
1		0960	Jim Searcy, 1400
1		1070	Ray Bair, 1200
1		1079	Al Romig, 1300
1		0953	Bill Alzheimer, 1500
1		0739	David Williams, 6421

1	0429	Ron Andreas, 2100
1	0630	Mike Eaton, 4010
1	1380	Warren Siemens, 4200
1	1380	Olen Thompson, 4221
1	1380	Mary Monson, 4212
1	1380	Vic Chavez, 4213
1	1380	David Larson, 4231
1	1380	Kevin Murphy, 4221
1	0801	Melissa Murphy, 4900
1	0622	Herb Pitts, 4400
1	0803	Jack Jones, 4600
1	1180	Pace Vandevender, 4700
1	0353	Mike Robles, 3800
1	0961	Bob Reuter, 1401
1	1434	Dave McVey, 1890
1	0661	Bob Parks, 4612
1	1110	Ray Tuminaro, 9222
1	0819	Gene Hertel, 9231
1	0437	Gregory Sjaardema, 9117
1	0660	Ron Sikorski, 9622
1	0805	Jeff West, 4911
1	1109	John Mareda, 9225
1	0836	David Sundberg, 9116
1	0439	Garth Reese, 9234
1	1169	Mike Furnish, 9322

1	9018	Central Technical Files, 8523-2
5	0899	Technical Library, 4414
1	0619	Print Media, 12615
2	0100	Document Processing, 7613-2 For DOE/OSTI