

UAMS, A STUDY IN SYSTEM ADMINISTRATION AUTOMATION

By

ROLAND JOSEPH STOLFA

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1986


Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1997

UAMS, A STUDY IN SYSTEM ADMINISTRATION AUTOMATION

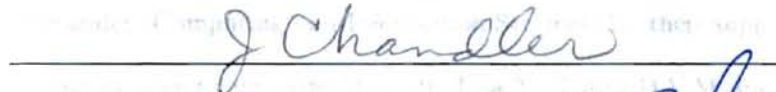
I sincerely thank my graduate advisor Dr. George Hedrick for the kindness, guidance, help and advice he has given me during the years it has taken me to write this thesis. Without the encouragement and help he has given me, the completion of this work would have been impossible.

I also sincerely thank Dr. Mayfield, Dr. Mitch Nelson, and Dr. John Chandler for working with me on this project.

My special thanks go to my wife, who has supported me throughout this project and whose love and encouragement has shown me the way through the most difficult times.



Thesis Adviser







Dean of the Graduate College

ACKNOWLEDGMENTS

I sincerely thank my graduate advisor Dr. George Hedrick for the kindness, guidance, help and time he has given me during the years it has taken me to write this thesis. Without the encouragement and help he has given me, the completion of this work would have been impossible. I also sincerely thank Dr. Blayne Mayfield, Dr. Mitch Neilsen, and Dr. John Chandler for serving on my committee.

My special thanks goes to Mr. Mark J. Vasoll for the encouragement and inspiration he has shown me during the course of developing the code associated with this thesis. Special thanks also go to Mr. Russ Smith (Mathematics Department), Mr. Rod McAbee and Mr. Clay Couger (College of Engineering, Architecture and Technology), Mr. Brad Barnes (College of Veterinary Medicine), and Mr. James Alexander (Computing and Information Services) for their support.

My respectful thanks goes to my wife Mrs. Dr. Lisa Y. Tresp, D.V.M and our daughter Ms. Victoria F. Stolfa and our son Mr. Kevin M. Stolfa for all the love, encouragement and support they have given me in writing this thesis.

TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION/HISTORY	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Organization	3
2. LITERATURE REVIEW	4
2.1 Asmodius	4
2.2 ACMMAINT	4
2.3 GAUD	5
2.4 newu	5
2.5 Uniqname	5
2.6 SHARE II	6
2.7 simon	6
2.8 GeNUAdmin	6
3. UAMS TODAY	8
3.1 The Database	8
3.1.1 The Database Description	8
3.1.2 Global Fields	10
3.1.3 Local Fields	11
3.1.4 DB Views	11
3.2 URI's	12
3.2.1 Automatic Rights	12

3.2.2	Granted Rights	12
3.2.3	Expiring Rights	13
3.2.4	Anti Rights	13
3.3	Rights Alias File	14
3.3.1	Slave side	14
3.3.2	Master side	15
3.4	Shell Strings	15
3.5	Universal Computing Identifiers	15
3.5.1	Background	15
3.5.2	Creation Heuristics	16
3.5.3	Special Case Users	17
3.6	Security Issues	17
3.6.1	URI's & Access Control	18
3.6.2	Configuration Spoofing	18
3.7	Initial Password Creation	19
3.7.1	Background	19
3.7.2	Creation Heuristics	19
3.8	Runtime Configuration	20
3.8.1	The UAMS Master/Slave Configuration File	20
3.9	Server/Client Model	21
3.10	Clients	21
3.10.1	UNIX	21
3.10.2	POP	23
3.10.3	Card Key Lock System	23
3.10.4	Novell	23
3.11	Master/Slave Server Model	24
3.12	Master to Slave Messages	25

3.12.1	Batch Load	25
3.13	Slave to Master Messages	27
3.13.1	User Query	27
3.13.2	User Remove	28
3.13.3	Voting and Expiration of Records	28
3.14	Summary	29
4.	CENTRALIZED FLAT NAMESPACE SYSTEM	31
4.1	What is it?	31
4.2	Benefits	31
4.2.1	Provided to the Individual Users	31
4.2.2	Provided to the System Administrator	32
4.2.3	Provided to the University	33
4.3	Costs	33
4.3.1	Preliminary Costs and Tasks	34
4.3.2	Continuing	35
4.4	Summary	36
5.	BLACK BOX DESCRIPTION OF A CFNS	37
5.1	Minimal Data Structures Needed	37
5.1.1	People Database	37
5.1.2	Association Database	37
5.1.3	Cross Reference Database	38
5.2	Supported Client/Server Functions	39
5.2.1	Make Return Set	39
5.2.2	Query By	39
5.2.3	Return Set	40
5.2.4	Vote For Set	40

5.2.5 Merge Ids	40
5.2.6 Create A New User	41
5.3 How it all Fits Together	42
5.4 Other Issues	42
5.4.1 Data Base Inconsistencies	42
5.4.2 Purging the Data Base of Old Records	43
5.5 Summary	43
6. PROPOSED FUTURE WORK AND CONCLUSIONS	44
6.1 Future Work	44
6.1.1 UAMS-lite	44
6.1.2 Real Time Access to the Master UAMS Site	44
6.2 Conclusions	45
BIBLIOGRAPHY	46
APPENDIX A: GLOSSARY	48

LIST OF TABLES

Table	Page
3.1 Logical Record	8
3.2 Physical Records	9
3.3 Global Fields of the Database	10
3.4 Departmental Fields of the Database	10
3.5 Example UCI Generations	17
5.1 People Database	37
5.2 Association Database	38
5.3 Association Database, Example 1	38
5.4 Association Database, Example 2	38
5.5 Cross Reference Database	39
5.6 Return Data Set Fields	40
5.7 Merge RPC Parameters	41

LIST OF FIGURES

Figure	Page
3.1 Sample datafeed for UNIX client system	22
3.2 Master/Slave and Server/Client Relationships	24
3.3 User Query Event Sketch	26
3.4 Master Side Processing of User Query	27
3.5 Sample Slave to Master VOTE Data Feed	28

CHAPTER 1

INTRODUCTION/HISTORY

1.1 Background

Prior to the development of The User Data Base System, UDB, at Oklahoma State University the two support staff members at Oklahoma State University Computer Science Department had to maintain over 40 machines manually. This included all the normal system administration of hardware, backups, mail, and news, as well as account creation. They often found themselves in search of a machine on which a user wanted an account, logging-on, creating the account, then finding the user and informing them of the initial password. In many instances, several days might pass between the request and the creation of the account due, in part, to playing "telephone tag." Another scenario would involve the creation of enough accounts for an entire class. This typically implied that there were no initial passwords for those students.

As time passed, access to the two terminal labs that the department runs for graduate students also became a chore. This involved keeping track of many keys to the labs in question. Monitoring the labs, and the equipment contained therein, became enough of a worry that a set of magnetic strip readers and a card key access system was installed on six doors in the department to monitor the use of the rooms.

With the advent of Network File System (NFS) hosts/servers, file sharing amongst one user's account on several machines became a problem due to NFS needing the numeric user identifiers to be the same on all systems. This involved merging many separate password files, sorting out what id's were in use on what machine, and then changing the numeric user id's on all machines across the network.

In short, there was no real solution to the user account creation, deletion, and management problems for a university. This situation lead to the original development of The User Data Base System, UDB, in 1987. Further refinements lead to the presentation on UDB at the Large Installation Systems Administration Conference in 1990 (available as OSU-CS-TR-90-04) [SV90b].

After the first paper on UDB, several other colleges and departments within OSU decided to participate in the types of services UDB was providing. However, to extend UDB to that domain would have required all departments that wanted to participate share one database on one host. The OSU Computer Science Department found itself in need of a system of providing database services to a majority of the campus. A system was needed that could maintain a campus wide flat name space, for both logins, or Universal Computing Identifiers (UCI) as we call them, and numeric user id's (NUID), allowing separate administrative entities access to only those pieces of information that directly relate to their organization. The result of this was the User Account Management System (UAMS) [Sto93].

As it stands today, UAMS (the term that describes the current merger of UDB and UAMS) is a centralized database of user id's, supporting several departments and colleges within the university in a distributed manner.

1.2 Problem Statement

Historically, the Computing and Information Services group on campus had not issued personalized user id's (or PUDs) to students. However, in 1992, CIS reversed themselves on this topic. In the ensuing three years, the number of PUDs handed out by CIS has exploded. In 1995, a new fee, the Student Technology Fee, was established as a tax on the students to supply services for the students as directed by the students. One of the side effects of this is that the students wanted to have email accounts for all students starting in the fall of 1995. The CIS department now provides PUDs to all incoming freshmen, in part to meet their obligation to the Student Technology Fee Committee. In order to do this, CIS intends to support a Centralized Flat Name Space (CFNS) system for the entire campus. Using this system, all users would have exactly one user login name across all platforms on campus.

However, UAMS has UCIs for virtually all OSU users (approximately 15000 at this time). In addition, the UAMS system has many of the features that CIS finds desirable for their support of computing on campus. As a result, CIS has offered to merge their PUD database with the UCI

database maintained by UAMS in the Computer Science (COMSC) department if COMSC develops the code to support this merger on the UAMS side and if COMSC will advise CIS in the development of their part of the code to support this merger. The resulting CFNS will be accessible by all of the current UAMS supported departments, and will allow CIS to expand the CFNS concept to the entire university.

The problem, and the focus of this thesis, is the result of the joint effort between the COMSC Department and CIS here at Oklahoma State University to implement a centralized flat name space system for the entire campus, based on UDB and UAMS, and further to deal with the merging of the two databases of information (PUDs -vs- UCIs). One of this thesis' goals is to describe how such a modified UAMS system would work, whom it would benefit, and how much work it would require to achieve the goal of a centralized flat name space system.

1.3 Organization

The thesis is divided into the following chapters:

- Chapter 2: A discussion on previous work related to user account generation and maintenance software.
- Chapter 3: Internals of UAMS today, given as an example of a working distributed flat name space system.
- Chapter 4: Centralized Flat Name Space Systems Described.
- Chapter 5: Black box description of a CFNS.
- Chapter 6: Conclusions and Proposed Future Work.
- Appendix A: Glossary.

CHAPTER 2

LITERATURE REVIEW

National refereed publications in this area started in about 1988, as that is when the Large Installation System Administration Conferences of the USENIX Association began. Several other account maintenance systems have been reviewed. Some of their high points are summarized below.

2.1 Asmodius

In Asmodius [EVS88], a distributed database system and reliable datagram socket connection protocol is described to perform some of the same functions as UDB. Further, the implementors of Asmodius chose to rewrite and replace several of the standard operating system utilities, such as *passwd*. These modified utilities then communicate to a centralized daemon which would in turn modify the central database and communicate the changes to other daemons running on those systems under Asmodius' control. These daemons then modify their local copy of the database and actually change associated operating system specific files, such as */etc/passwd*.

However, Asmodius utilizes daemon programs running on both the server and the clients that rely on TCP/IP networking facilities. Although these facilities are gaining popularity here at Oklahoma State University, not all hosts at OSU have TCP/IP, hence this approach was not usable.

2.2 ACMAINT

In ACMAINT [CKCS90], a central database was presented to allow a single system administrator to manage computer account creation across a heterogeneous set of computers. ACMAINT, like Asmodius, was designed around modifying several of the operating system utilities (*passwd*, *chfn*, and *chsh* to name a few) to communicate with a centralized database. This database then transmits the "change" information to a daemon running on each network attached host under ACMAINT's control to perform the change.

In ACMAINT, the "change" daemons do not keep track of the entire database, nor do they keep a view of the database appropriate to their system. Instead, these daemons are strictly in charge of

processing change requests. This is the major difference between Asmodius and ACMAINT.

However, ACMAINT utilizes daemon programs running on both the server and the clients that rely on TCP/IP networking facilities. Again, as these facilities are not universally present in the OSU environment, ACMAINT was inappropriate for use here at OSU.

2.3 GAUD

In GAUD [Urb90], a central database is accessed by many hosts over Remote Procedure Call (RPC) protocol to allow access by the various offices that might allow or deny access to a particular user to a particular machine. Furthermore, GAUD has as design goal the maintenance of extra data to assist in various financial accounting procedures. In addition, GAUD maintains a list of where a users home directories reside on a distributed NFS file server farm.

However, GAUD suffers from the reliance on the availability of source code to the operating system, an item not all universities have. Furthermore, at OSU, RPC is not available on all hosts, hence its unsuitability.

2.4 newu

In newu [SV90a], describes a functionality that is already in UDB; ie. the ability to create and delete accounts on a foreign host. In addition, newu manages disk quota issues for the users it administers.

However, newu suffers from the same problem UDB had, in that it only worked within one administrative entity. Further, newu was developed to deal with the workload associated with adding one user to many machines. It does not provide any mechanism for dealing with groups of adds and deletes, such as seen in a university environment around the change of semester enrollment flux.

2.5 Uniqname

In Uniqname [DLM90], a system of merging existing accounts with a 'global view' is presented. This is the result of several different computer systems, and incumbent user populations, existing prior to the development of Uniqname. Uniqname also attempts to address a few divergent issues, such

as X.500 email addresses, Kerberos authentication, and AFS security systems.

As UDB started with a unified 'global view', Uniqname offered a solution to something that was not a problem in our case. In addition, it presents a solution to several problems that UDB did not even attempt to address, such as that of preferred mail box address.

2.6 SHARE II

In SHARE II [BGMR94], an overall object oriented resource control system for areas such as CPU utilization, memory use limits, and disk quotas is presented. It controls many aspects of system administration that through source code modifications to both the kernel and several operating system utilities. In addition to these areas, SHARE II also deals with user account management.

As not all sites here at OSU have the source to the operating system, this is not a viable alternative.

2.7 simon

Simon [Fin92] presents a system similar to UDB that relies on a commercial database system. Simon receives data from the registrar, from payroll processing, and has special case "guests" manually entered. It then produces a resulting view of who should be on a system. In addition, this system manages some aspects of a charge back system.

As these things are undesirable here at OSU, this system was not a viable alternative to UDB.

2.8 GeNUAdmin

In GeNUAdmin [Har94], Harlander describes an automatic system administration tool used to control various operating system parameters, tuning variables, user default variables, file system mounting options, and user accounts across a network of heterogeneous computer platforms. One of the design goals of GeNUAdmin is to allow ease of administration by performing various consistency checks to both the operating system configuration and the user population.

Since it involves many more aspects of system administration than UDB, much of GeNUAdmin can be ignored. The user administration aspects of this paper describes replacement code for several

system utilities, such as 'passwd'. Since this is something that UDB was explicitly designed to avoid, this system was not a viable alternative to UDB.

CHAPTER 3

UAMS TODAY

This chapter describes the internal organization and the operating principals behind UAMS as it exists today. This chapter is included so the reader will have a better understanding of both the history and the functionality that is being sought.

3.1 The Database

3.1.1 The Database Description

Conceptually, the database is a relational database with a single relation, consisting of the fields shown in Table 3.1. These fields have evolved through the development of UAMS and the uses to which it was put. Due to the operational use of these fields however, the logical record is divided into six physical records (Table 3.2).

These six physical records are then stored in a hashed database. Originally this was done using Ken Thompson's DBM (as supplied with UNIX¹). However, as the size of the database grew, it exceeded DBM's capabilities. At that time, the Gnu Project's GDBM was chosen to replace DBM. For each of the physical records (shown in Table 3.2), a DBM *key=value* pair is generated. Typically the first character of the physical record name is prepended to the NUID to form the *key*. The *value*

¹ UNIX is a trade mark of X/Open.

osu_id	Student/Faculty identification number (unique)
issue	Student/Faculty ID card issue number
fullname	Full name of the individual, as defined by the university
uci	Universal Computing Identifier (unique)
nuid	Preferred Numeric User ID for NFS (unique)
passwd	Clear text password
epasswd	DES encrypted password text
major	Student department affiliation
auto	Automatic enrollment rights
granted	Manually granted rights
comment	Comment field
lupdate	Last update time of this record

Table 3.1: Logical Record

General Record	
osu_id	Student/Faculty identification number
issue	Student/Faculty ID card issue number
fullname	Full name of the individual, as defined by the university
uci	Universal Computing Identifier
nuid	Preferred Numeric User ID for NFS (DBM key)
passwd	Clear text password
epasswd	DES encrypted password text
major	Student department affiliation
lupdate	Last update time of this record

Automatic Record	
nuid	Preferred Numeric User ID for NFS (DBM key)
auto	Automatic enrollment rights

Manual Record	
nuid	Preferred Numeric User ID for NFS (DBM key)
granted	Manually granted rights

Comment Record	
nuid	Preferred Numeric User ID for NFS (DBM key)
comment	Comment field

OSU Id Map Record	
osu_id	Student/Faculty identification number (DBM key)
nuid	Preferred Numeric User ID for NFS

UCI Map Record	
uci	Universal Computing Identifier (DBM key)
nuid	Preferred Numeric User ID for NFS

Table 3.2: Physical Records

osu_id	Student/Faculty identification number (Unique)
issue	Student/Faculty ID card issue number
fullname	Full name of the individual, as defined by the university
uci	Universal Computing Identifier (Unique)
nuid	Preferred Numeric User ID for NFS (Unique)
major	Student department affiliation
auto	Automatic enrollment rights

Table 3.3: Global Fields of the Database

passwd	Clear text password
epasswd	DES encrypted password text
granted	Manually granted rights
comment	Comment field
lupdate	Last update time of this record

Table 3.4: Departmental Fields of the Database

is typically the plain text contents of the field; in the General record case, the *value* points to a structure that contains the listed fields.

In the operation of this relational database, an UCI to OSU_ID record exists for each user, as does a General Record. Each user who has any enrollment data has a corresponding Automatic Record. A Manual Record exists for each user who has special rights on the server. The COMMENT record exists for the system administrators to keep a textual tag to be associated with the user.

3.1.2 Global Fields

The data analysis concluded that all slave UAMS sites across campus must share some part of the the global database of logical records maintained on the master site. The slave UAMS sites would treat this data as read-only, allowing the master UAMS site to overwrite these fields when needed. The global fields are treated as read-only on the master UAMS site after the initial creation of the users record. This is because the entire list of UCIs and NUIDs are kept unique on the master UAMS site. Once generated uniquely on the master site, these global data fields can be transmitted to any of the slave UAMS sites while still guaranteeing the data integrity; ie. no duplicate UCIs or NUIDs (Table 3.3).

3.1.3 Local Fields

The local data fields are unique to each administrative UAMS site. They are not transmitted either from the slave to the master site or vice versa. This allows each UAMS site to have control over the special case users without infringing on any other UAMS site (Table 3.4).

3.1.4 DB Views

As of the development of UAMS, the database was further modified to be distributed between a master and several slave UAMS servers. The master server maintains the uniqueness of UCI's and NUID's, receives the data feed from the central university database, and handles requests for new users from slave servers. Slave UAMS servers contain a view of the master UAMS database, appropriate for the department that is running the slave server. Slave servers are also where a departmental UAMS administrator configures the different clients.

This arrangement allows different UAMS administrators to configure their respective views of the database to affect their department in a manner that is consistent with that department's wishes. It also allows for different, sometimes conflicting URIs, or Universal Rights Identifiers as we call them, to be used within different slave UAMS servers without interfering with the other departments.

One consequence of this splitting of each logical record into a global and local part is that each administration is able to set the default initial password, while maintaining the same UCI. This helps maintain some level of security among UAMS hosts. Using this arrangement disables one user, knowing their UCI on one UAMS administered host, from logging in ad-hoc to other UAMS administered hosts based purely on the knowledge of the original password. It also implies that an individual user's account information on one system has minimal use on another, as the initial password is different between UAMS administrations.

Each administrative UAMS site also has a separate and unique GRANTED right field for each user in their system. This allows each site to specify unique (and possibly conflicting) URIs to give access to different clients (hosts, etc.). On the master UAMS site, the GRANTED right field is also

used to select those special case users, such as “root”², who need to visit a slave UAMS site in addition to those users destined to go to the slave site because of enrollment information. However, since the GRANTED right field on the master UAMS site does not go with the record to the slave site, the slave UAMS site never sees that URI.

Another result of this data analysis is that each department is allowed their own comment field (COMMENT) for each user. That way, any comments on a user are held in the confidence of the commenting department.

3.2 URI's

3.2.1 Automatic Rights

The data held in the “auto” rights field contains information related to enrollment and employment. Contained in a separate physical record, this field is a list of all relevant courses this user is enrolled in this semester. In addition, all employees of the university that happen to be in the UAMS system have are given a dummy enrollment record to indicate the department for which they work, and what kind of an employee (student, faculty, etc.) this user is. All data in this field expires automatically when UAMS receives new data from the central university database.

These rights take the form of a comma separated list of text strings. This field of the logical record was split into a separate physical record because the data contained in it must to be updated each time a new university database data feed is received. Having this as a separate field simplifies the update process by allowing the system to simply remove all automatic records in one pass. The program that receives the enrollment data can simply reload the course enrollment data by regenerating all the automatic records.

3.2.2 Granted Rights

The data held in the GRANTED rights field are special case rights given on an as needed basis. Such things allow users such as “root” to have an account on all machines without concern that their accounts could be deleted automatically at some point in the future. In addition, there are

² On most UNIX systems, there is a special account used for system administration tasks. It is typically called “root” and is a privileged account.

always special case users who need an account (or access to a room via the card key access system) for a specified period of time but they would not otherwise be granted that right. This field of the logical record was split into a separate physical record because it was found that there were very few of these compared to the number of total users in the database (10:1 on average).

3.2.3 Expiring Rights

Any GRANTED right may have an expiration date of the form “-YYYYMMDD” appended to it. Upon a periodic ³ scan of the database, all GRANTED rights that have reached their expiration date are removed from that user’s record.

3.2.4 Anti Rights

As in almost any other university setting, students will be students. As UAMS was applied to an ever larger body of students, it was inevitable that a student would need to be prohibited from using a machine due to an infraction of the rules. Shortly before this became necessary the author developed the anti-right. With this GRANTED right, a user could be excluded from a machine, regardless of other rights.

For example, the user “foo” is to be prohibited from using machine “A”. However, “foo” is granted access to that machine due either to a class enrollment right (in an Auto field in this user’s record), or to their Major (if they are a guest on that machine, indicated by a URI in this user’s GRANTED field of, say, “A”, the GRANTED right “A” is simply removed). If this was done, the next time UAMS received this user’s enrollment data from the central university database, the Auto field enrollment right would return. At this time, the Major code would be restored also. As these are not solutions, the anti-right was developed. In the case of “foo”, a GRANTED anti-right would be given as “!A”. In fact, in order to lock “foo” out for a specified period of time, an expiration date could be added, giving an expiring anti-right of “!A-YYYYMMDD”.

These anti-rights do nothing abnormal to the record of the user. Instead, they alter the list of users selected to go to a site, “A” in this case, to exclude this user. If this same user has other

³ This process is typically activated once a day, but may also be initiated on demand.

GRANTED rights, for instance this user has an account on the same department's POP server, their POP server rights are unaffected.

3.3 Rights Alias File

The Rights Alias File, or RAF, is the central control mechanism within UAMS. The definitions within it control which records are selected to go to which other clients. It also defines which URIs may be given to a user manually in the GRANTED right field of their record.

As may be guessed, looking at the manual page on RAF that comes with UAMS [Sto96], all blank lines and lines that begin with “#” are comments. All other lines are broken into three colon “:” delineated fields. The first field is an alias for all lines referenced with the same alias. The second field is the list of explicit URIs are to be used as a selection criteria for this alias. The third field is an optional command field associated with this alias.

In short, the rights for a particular alias indicate all records that are to be extracted and passed thru the optional command to generate the information necessary for the client that the alias represents.

It is possible to maintain a list of users with a particular URI without actually using the list in any alias line. This is done by leaving the first and third field blank and listing all the URIs of interest in in the second field.

3.3.1 Slave side

On a Slave server, the RAF is typically concerned with mapping which URIs into which client side programs. Each installed Slave server comes with several client side programs. Among the more useful ones are “gen_rolls,” [Sto96] a program that prints out a class roll sheet for an instructor to use when determining the login name and password for each student in a class. Another useful program is “gen_unix,” [Sto96] which prepares a data feed to a (possibly remote) site for the creation of user accounts on that site.

3.3.2 Master side

On a Master server, the RAF is generally a spartan affair that simply maps which classes of users go to which slave site. A program called "gen_asd" [Sto96] does this job. When it is finished, a data feed is shipped to a given slave site that contains the "global" part of the master site's database for most of the records that the given slave site has.

3.4 Shell Strings

The Bourne shell and awk play a big part in UAMS. However, taking advantage of the DBM database requires a clean method of getting the data from the DBM to the shell and vice versa. The Bourne shell has environment variables of the form "name=value". It also has a method of evaluating a string of "name=value" pairs where each pair is separated by a semi-colon (i.e. "eval"). Hence, the UAMS program "udbget" [Sto96] extracts the data from the DBM into a string of "name=value" pairs with each of the requested fields semi-colon separated. Further, the UAMS program "udbput" [Sto96] parses the same type of string and inserts the updated data back into the DBM.

3.5 Universal Computing Identifiers

3.5.1 Background

A major part of the emphasis of the original UAMS was devoted to the generation of "meaningful" user id's. Previously, all user accounts were generated in the form of "prefix_count" where each user shared a "prefix" with all classmates and had a unique "count". For example, a file structures class with three students would have three user log on account names generated as "fs1," "fs2," and "fs3." This complicates the administrative job by hiding the person behind a "fs1" account (as the faculty member assigned the accounts to the students). In addition, some users had multiple accounts. This was the result of one student enrolled in several different classes, each with a different computer account for the programming assignments for that class on the same machine.

3.5.2 Creation Heuristics

More meaningful user names were desired. A few experiments with the enrollment data led to the following extensible heuristic ⁴.

1. First, all illegal characters are removed from the full name. These include such things as hyphenation and punctuation. Then the name is converted to lower case.
2. If a person's last name is greater than seven (7) characters and their first name is greater than three (3) characters and less than or equal to seven (7) characters, then their first name is designated as "word1" and their last name is designated as "word2". Otherwise, their last name is designated as "word1" and their first name is designated as "word2."
3. Following is the list of checks for uniqueness. When one of these database probes determines that the UCI is not found in the database, then that UCI is assigned to this user. The first one to be chosen ends the algorithm.
 - (a) Up to the first seven characters of "word1."
 - (b) The first character of "word1" followed by up to the first six characters of "word2."
 - (c) Up to the first six characters of "word1," followed by the first character of "word2."
 - (d) The first five characters of "word1," the first character of "word2" and the first character of this user's middle name.
 - (e) The first seven characters of "word2."
 - (f) The user's initials; i.e. first character of first, middle, and last names concatenated.
 - (g) The first initial, their middle initial, and up to the first 5 characters of their last name.
 - (h) Up to the first six characters of "word2" followed by the first character of "word1."
 - (i) Up to the first five characters of their first name followed by their middle initial and then their last initial.

Name	3A 3E	3B 3F	3C 3G	3D 3H	3I
Bogus Jay Serendipity	bogus serindi	bserend bjs	boguss bjseren	bogussj serendb	bogusjs
Bogus Serendipity Jay	jay bogus	jbogus bsj	jayb bsjay	jaybs bogusj	bogussj

Table 3.5: Example UCI Generations

Table 3.5 shows several of the possibilities for a pair of fictitious users.

3.5.3 Special Case Users

To handle the special case users such as “root” required several special considerations. First, these users did not have OSU_IDs, any enrollment data, and quite seldom a FULLNAME. To cover these cases, as well as the case of the occasional guest account (or odd software package) that did not have an OSU_ID, a simple heuristic was formed: taking the proposed UCI (say “root”) and using a “+” prepended to the UCI as the OSU_ID. Thus “root” would have the OSU_ID of “+root”.

This simplifies some areas of system administration. For instance, all of the users with a plus in their OSU_ID field have no OSU student or staff id. Therefore they need not be selected for loading into the card key lock software⁵.

Also, they are typically what we call “mechanical accounts,” ie. they come with an operating system and do not have a physical person behind them. So when scanning the password file for accounts to set no-login, the system administrators can use UAMS as an aid in this process⁶.

3.6 Security Issues

As in any large database project of this sort, security issues that are in direct contradiction to the general purpose of the project arise; namely providing the user data. In UAMS, there are several types of security imposed on the access to the data to make the data fairly secure.

⁴ It is extensible, but has served our needs for the last two years.

⁵ As described in [SV90b], the Computer Science Department runs a card key lock system on several labs within the department.

⁶ This is not an enforced function of UAMS, merely an example of using the data UAMS maintains.

UAMS assumes that each user on the system belongs to a group of their own. Each user id number and their group id number are identical (for any user), and these numbers are unique amongst all others on the system. The following security discussion *REQUIRES* that this be so.

3.6.1 URI's & Access Control

When all of the programs, shell scripts, and configuration files of a new master, slave, or client UAMS code set are installed, it is assumed that these files are "chown(1)"ed to a particular user. In the case of the UAMS master and slave code, this user is assumed to be "udb." The installation script provides all the "correct" file permissions for every file installed. Once all the files are owned by the assigned user, the programs and data are secure from tampering.

Access is granted to the data contained in the database via URI's given the user in question. If the default *UDBPRIV* is selected (during configuration), a user must possess the "udb_priv" URI before being given access to the data. This privilege is typically, and most securely, a GRANTED right, outside of the conventional name space of all other URIs on the system thus avoiding accidentally giving a user a URI that will allow them access to the database.

Once a user has attempted to run a C program component of UAMS, a library module written in C, *allow.c* [Sto96] to be precise, is used to determine if that user has been granted the correct URI. If the user is allowed, access is granted. If the user is not allowed, the attempt to gain access is logged and the user is refused access to the database.

None of the shell scripts within UAMS access the database directly. They all rely on C programs to perform the access on their behalf. Thus, using "allow.c" in the C programs secures all the data contained in the database.

3.6.2 Configuration Spoofing

As discussed in Section 3.8, the configuration can be changed by defining one simple environment variable. This does not, however, change the URI needed to gain access to the database. This mechanism is configured into *SRCROOT/h/udb.h* during the configure/compilation stage in order to prevent this kind of security attack.

3.7 Initial Password Creation

3.7.1 Background

In the past, accounts were typically generated without passwords or a single initial password for the whole class. This posed several problems. Sometimes a malicious user would log onto as many of the accounts as possible and set a password on that account. This prevented the rightful user from logging in. Another case that has occurred is where a new graduate student has been given an account for the class he or she is teaching. These accounts were also generated without passwords. However, some of these new students didn't know that a password should be set, thus allowing anyone to log into their account where such things as grades, student id to student name mappings, etc. may be found.

3.7.2 Creation Heuristics

At account creation time, UAMS generates a random password and associates it to the new account.

The generation of the password proceeds as follows:

1. A random number is generated and divided modulo the number of words in the word list. This word is retrieved and is called "word1."
2. A random number is generated and divided modulo the number of separator characters. This character is retrieved and is called "word2."
3. Another word from the word list is chosen, as in "1" above, and is called "word3."
4. The password is generated by concatenating "word1," "word2," and "word3."

Several examples of the result of this algorithm would be: sue1cat bob\$dog imp!yack

This word is then passed to the DES encryption algorithm that comes with UNIX to generate the encrypted password, and this is also associated with the account. On export versions of Unix, this will not work, due to the non-DES algorithm being installed in the system libraries.

Both the plain text version of the password and the encrypted version are saved in the database. Only the plain text version is truly needed; however, the regeneration of the encrypted version is a very time consuming process (using the DES algorithm). It is for this reason that we generate the encrypted version only once and then save it for all future uses.

Users are encouraged to change their password as soon as possible, so that it no longer matches the UAMS database. Their current actual password is not stored, only an initial value for the creation of a new account is stored.

3.8 Runtime Configuration

This section covers the run-time configuration process once a production environment is set up.

3.8.1 The UAMS Master/Slave Configuration File

The file `udb.cfg` [Sto96] contains the central configuration file in either a master or slave UAMS site. It is a plain text file and contains comments that outline what the variables contained in it are for, as well as configuration comments that allow the program “subst”⁷ to provide reasonable defaults for all variables.

In shell scripts

During the running of any of the shell scripts within the UAMS package, “`udb.cfg`” is read first. This is the whole purpose in editing in the path to `UDBROOT/config/udb.cfg` during the configure/compilation stage for all the shell scripts. This is also where the “`PATH`” environment variable is set. As soon as any shell script is started, the assignment of “`PATH`” in this file overrides any previous value that either the user or any external wrapper script may have set.

In each script, there is a variable “`UDB_NAME`” that is set to the name of the shell script that is running, *BEFORE* “`udb.cfg`” is included into the running shell script. This allows for user defined triggers to be placed within “`udb.cfg`” to do special things. One such thing, of great use during debugging phases of the system, is to “`set -x`” (based on the `UDB_NAME` of the module to debug).

⁷ The shell script program “subst” is a part of the USENET news reading package “C-News”.

In C programs

Within the UAMS package, in `cfgstr.c` [Sto96], there is a library C function that parses most of the data from `UDBROOT/config/udb.cfg` and makes it available for use within the C functions. Any procedure may access the variables defined and set in "udb.cfg" in much the same way that shell scripts do.

In each program, after the user has been validated as authorized to run the program in question, "udb.cfg" is read in and parsed to give the program the same definitions.

The UDBCFCG environment variable.

In the previous two sections, `UDBROOT/config/udb.cfg` has been treated as a fixed path. In fact, it is not. The environment variable `UDBCFCG` may override the definition of `UDBROOT/config/udb.cfg` at any time. `UDBCFCG` must point to a file that contains all the same variables as the default one (in `UDBROOT/config/udb.cfg`), with possibly different right hand sides, to make the system run.

Both shell scripts and C programs used in UAMS obey this convention.

3.9 Server/Client Model

A server/client relationship exists between any UAMS site and a system that is administered by the owner of the UAMS server. In this system, any server, either a master UAMS or a slave UAMS, may provide a data feed to a client system. The format of the data delivered to the client is client specific, and typically is used to generate some end product specific to that client.

3.10 Clients

Since there are several UAMS clients, this section concentrates on the largest example, the UNIX client. The other clients available as UAMS clients will also be briefly discussed.

3.10.1 UNIX

This collection of shell scripts reads a data feed from a UAMS master or slave site (Figure 3.1) and generates the correct password file, group file, and login directories for each new user described

```

...
uci:epasswd:fullname:granted:major:auto:nuid
...

```

Figure 3.1: Sample datafeed for UNIX client system

in the data feed. It also deletes from the password and group files all users no longer in the data feed.

Furthermore, all URIs are propagated to all client UNIX systems from their administering UAMS server to allow the generation of the URI database for each host. This allows each departmental machine to make use of all of the enrollment data, the major code, and all of that departmental UAMS server's unique URIs for its own purposes. One of the uses of the URIs that has been implemented at OSU is the ability to run various programs (similar to access control lists). Another is the automated maintenance of mailing lists.

In addition, the UNIX client can provide the following services:

- Validate that certain users remain in the data feed. Without these special users, the data feed will not be installed.
- Validate that the data feed is for this site.
- Generate a URI database of which users have what rights on this system.
- Generate from the URI database, AT&T System V "cron.allow" - "cron.deny" type security files, from any given URI on the system.
- Generate an additional /etc/group line based on any given URI.
- Generate mailing lists based on:
 - Any right that any user on the system has, generates a mailing list.
 - Any user that has a right outlined in a list, generates a mailing list.

- Interface to the Multi-channel Memorandum Distribution Facility (MMDFII) [III84] mail transport system mailing list configuration mechanism.

3.10.2 POP

The Post Office Protocol (POP) client came about because there was an interest within several departments to provide mail to personal computers. The POP system, as distributed with the RAND Corp. MH mailer, was chosen by some of them to fill this need. This system had to be configured for users, just as the UNIX hosts did. As POP uses a password file that is in many ways similar to the UNIX password file, this client required several minor changes to one of the existing clients, allowing it to run with minimal system administration.

3.10.3 Card Key Lock System

Although not a system that anyone can log into, the Computer Science Department uses an Identacard brand card key access system to log access to the department's various labs and facilities. It is included here to show that seriously non-standard systems can be supported in a clean and efficient manner.

3.10.4 Novell

The Novell⁸ version 3.11 client came about because of one of the other departmental UAMS administrators. Within the other department, a Novell network was used to link together several personal computers within a student lab. However, all of these computers had to be configured for users, with much the same information, just as the UNIX hosts that UAMS already served. As this is just another type of host, using unique login names and passwords. Consequently, a new client was written to provide the information.

After researching the Novell manuals and considering the options, the Novell client was written to generate a data file for the Novell user administration program "MAKEUSER." The UNIX shell script "gen_novell" [Sto96] would keep a list of the users currently authorized for a Novell site, compare that with what UAMS was giving it, and generate add and delete commands as necessary.

⁸ Novell is a registered trademark of Novell Inc.

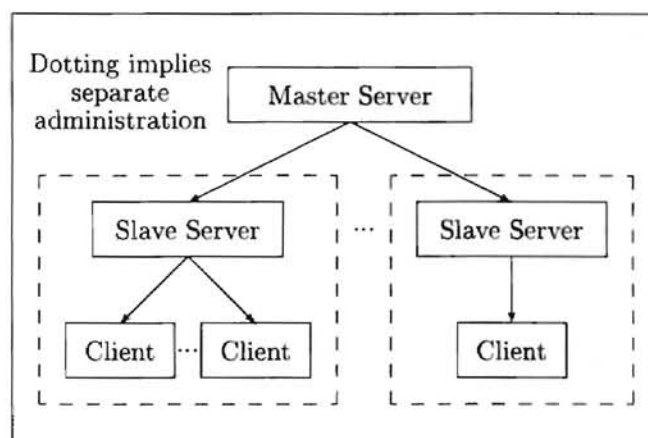


Figure 3.2: Master/Slave and Server/Client Relationships

This preserved the feel of the UNIX client, without having to write a program for a personal computer.

3.11 Master/Slave Server Model

In order to facilitate the sharing of data, a simple master/slave model was chosen (Figure 3.2). This represents the administrative association of the “global fields” of the UAMS databases among one another. Hence there must be a master UAMS site. This is the site that receives the enrollment data from the registrar. It is also the only UAMS site that can definitively assign UCIs and NUIDs. Whenever any slave UAMS site needs a UCI and NUID for a user new to it, then it must defer to the master UAMS site for the definitive information. The master UAMS site also maintains the master copy of the enrollment data for all users of all slave UAMS sites. This data is given to the master site after the participating UAMS departments authorize the master UAMS site to have this data.

The master/slave model allows the master UAMS site to select all of the records for a slave UAMS easily. This typically is based on enrollment data held in the Auto and Major records (for students). Additionally, to select all non-standard records (for such things as “root”, “uucp”, faculty, etc.), a specialized GRANTED right, unique to that slave UAMS departmental administration, is used.

To provide each participating department autonomy over their users, only the "global fields" are shared among the different UAMS sites. This implies that any URI given to a user on the master system as a GRANTED right is not propagated to any other UAMS slave site. This includes the specialized GRANTED right, as the GRANTED field is not in the "global fields" list of data being shared. This also allows each departmental UAMS to have overlapping (and possibly duplicated) GRANTED right URIs. Further, it allows each departmental UAMS to use the COMMENT field as they determine, without having to conform to some standardization scheme.

3.12 Master to Slave Messages

3.12.1 Batch Load

Once per day, the master site checks its database to determine whether it has been modified. If it has been modified, then the master site prepares a data feed for all of its slave sites. This is on the design assumption that all slave sites would be interested in any changes to the database. The data feed to the slave sites is timed to trigger a cascade of the data to the clients of that particular slave server site. The processing of "new" data feed is performed before the slave sites processing for data feeds from that slave site for its client sites (Figure 3.3).

The actual data feed is composed of the "global fields" of the database (Section 3.1.2). These are encapsulated with data to aid in the validation of the data in the data feed and electronically mailed from the master to each slave server site. Which records are sent is based on the selection criteria set out for the slave in the master site's RAF (Section 3.3).

This data feed is augmented by a password and an encrypted password at the slave site. In addition, the last updated time is set for this record. Through the record's history on this slave site, the last update field is updated every time the record is modified by either a "batch load" data feed or by the slave site administration modifying the data.

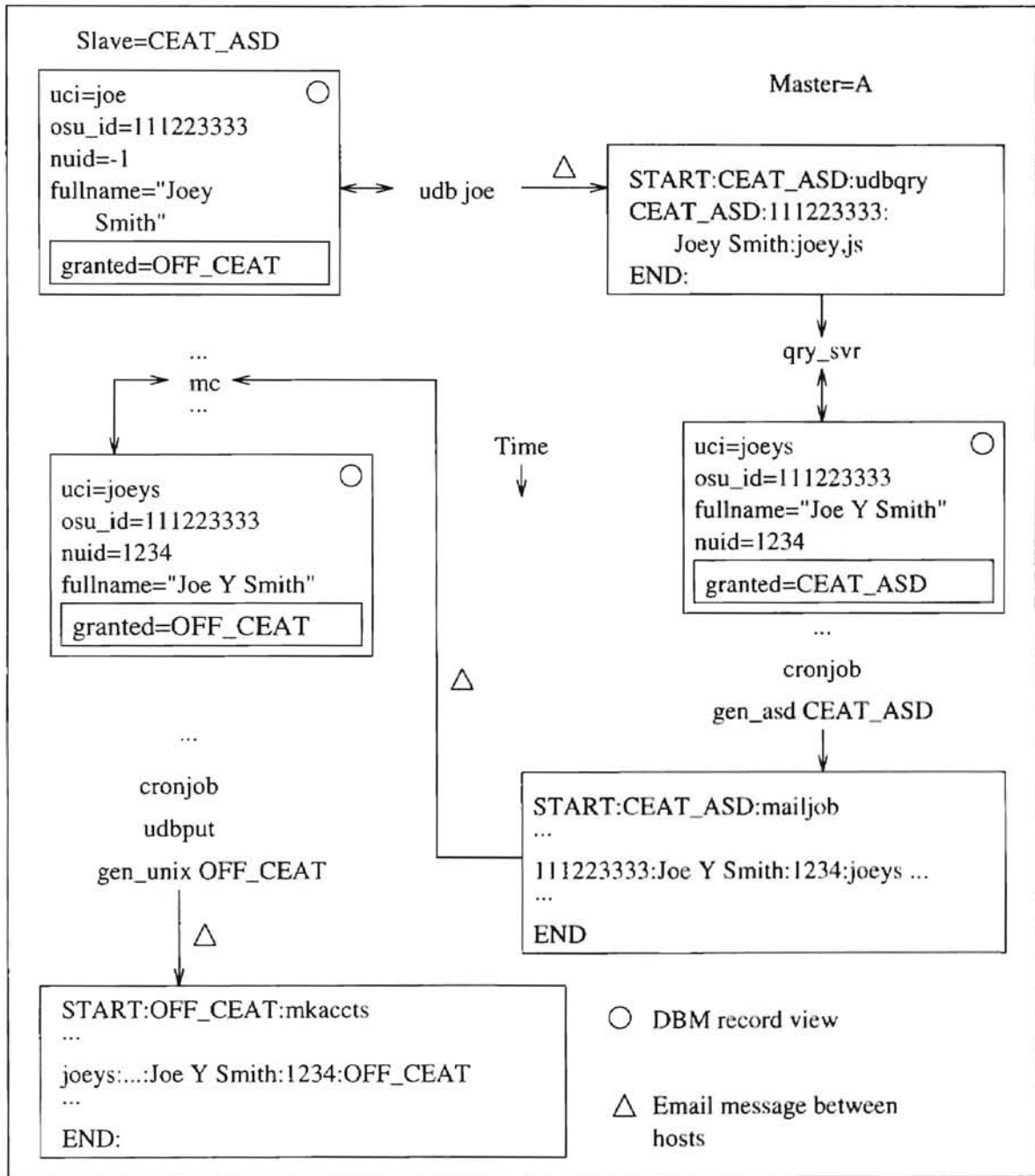


Figure 3.3: User Query Event Sketch

```

if (osu_id is NOT in the master site's DBM)
  if (some requested_id field is non-blank)
    if (requested_id.1 is NOT inuse)
      assign requested_id.1 to osu_id;
    else if (requested_id.2 is NOT inuse)
      assign requested_id.2 to osu_id;
    else if (requested_id.3 is NOT inuse)
      assign requested_id.3 to osu_id;
    fi
  fi
  if (uci field is still blank)
    create uci for osu_id using fullname;
  fi
  create nuid for osu_id from unused list;
fi
tag record for return to client for three weeks
update lupdate to today's date

```

Figure 3.4: Master Side Processing of User Query

3.13 Slave to Master Messages

3.13.1 User Query

The user query message is generated on a slave site when the administration of the slave site attempts to create a user's record for which there is no pre-existing data on that particular slave site. As shown in Figure 3.3, this message is generated, transmitted, and processed in real time when the slave site's administrator first requests the creation of new user data.

The record contains the OSU_ID, FULLNAME, and (optionally) the requested UCT's for this user. On the master site, the process outlined in Figure 3.4 is triggered by the receipt of the record and is used to fulfill the request.

At the end of this procedure, the (possibly) new record is given expiring GRANTED right on the master site that will select this record to go back to the requesting slave site via the rules in the master site's RAF file. This GRANTED right is typically set to expire in three weeks. This allows the slave site up to three weeks to receive notification of the creation of this new record. At the end of this period, the GRANTED right that forcibly shipped this record to the slave site is expired from the GRANTED field of the master site's record (see Section 3.2.3).

```

...
uci:osu_id:nuid
...

```

Figure 3.5: Sample Slave to Master VOTE Data Feed

The mechanism whereby the slave site keeps this record active on the master site is covered in Section 3.13.3.

3.13.2 User Remove

When a slave site finally needs to remove a user record from its site, it generates a “user remove” record. This typically is due to the slave site running a program that determines that the record in question does not have any propagation to any of that slave’s clients, and further, has not had any activity for a period of one year (see the manual page for “udbpurge(1)” [Sto96]).

On the master site, this record triggers a check of this user’s record to make sure no GRANTED right record on the master site continues to propagate this record to the slave site.

As in the case of the “user query” record, this record is generated, transmitted, and processed in real time. This record’s usefulness has been greatly augmented by the “vote” record, described in Section 3.13.3.

3.13.3 Voting and Expiration of Records

Once a month, all slave sites “vote” for those UCI, OSU_ID, NUID tuples that they have in use. On the master site, each “vote” is checked to determine whether the UCI, OSU_ID, NUID tuple is correct. If there is any problem, the record and its source are noted in a log file for human intervention. Otherwise, the LUPDATE field for the user record associated with the tuple is updated to the current date. On the master site, a record continues to exist until the following conditions are met:

1. There are neither any AUTO nor GRANTED rights associated with the user. This is typically associated with the person no longer being enrolled at the university.

2. The record has not been touched in a period of one year. This is indicated by a one year old LUPDATE field.

Special case users, like "root" and "uucp" are voted for by all slave sites every month, their records never expire at the master site. However, if a user's record meets the criteria above after dropping out of school, the record on the master is eliminated. This allows the user's UCI and NUID to be reused for the first new person who enters the university and who has a fullname that matches the generation criteria outlined in Section 3.5.

Each slave follows the same process. Each record is scanned to determine whether it meets the criteria outlined above. If it does, that record is removed from this slave site. As soon as the record is removed, this slave site no longer votes for that user. When all slave sites do not vote for this user, the one year clock starts. At the end of the year, the user's record is removed from the master site (as outlined above).

The listed procedure implies that once a person leaves the university, their record actually sits on the master site for a period of two years. The first year is spent awaiting the slave sites to expire the record on their local database, the second from the master site. This gives a user a period of two years in which to reenter the university and still have the same UCI and NUID.

3.14 Summary

The merger of UDB and UAMS provides to the OSU environment most of the features necessary for a CFNS. Among the most important is the concept of one user id (UCI) and one numeric user id (NUID) per user (as based on the OSU_ID number).

Unfortunately, one fact that prevents UAMS from filling the role of CFNS totally is that not everybody in the institution has a record in UAMS. This is because UAMS only receives records for those users whose departments have requested to be part of the UAMS project. Furthermore, as only those departments that have asked to join in the services that UAMS provides participate in the system, not all departments are served.

With the creation of CFNS services at the institution level, the central goal of one UCI and one

NUID per OSU_ID can be achieved. The central administration of the institution can simply assign one UCI and one NUID per OSU_ID for all members of the institution. In addition, a centrally maintained CFNS can provide CFNS services to a wide variety of platforms given a standardized protocol to many departments. This then is the goal of UAMS and this thesis.

CHAPTER 4

CENTRALIZED FLAT NAMESPACE SYSTEM

4.1 What is it?

A Centralized Flat Namespace System is a mechanism whereby all users of an enterprise's computer systems has exactly one unique login name. In the case of most computer operating systems, this is extended to include one unique numeric id also. These two pieces of data are linked to some part of the user's university data, such as employment records or enrollment data.

Over the life time of the user's association with the university, the unique login name and numeric user id are *stable*, that is they do not change without good cause ¹.

In all database views of the university database that include the unique login name and numeric user id, the *same* data is shown, to all users. Further, there exists at most one view of the data where either of these fields will be *changeable* associated with either becoming enrolled for the first time or upon employment by the university.

As has been covered in the previous chapter, UAMS was designed to solve many of these problems for the cooperating departments. However, viewed in the larger sense of a CFNS for the entire university, many of the problems that UAMS solves are duplicated with good reason.

4.2 Benefits

The benefits of having a CFNS installed and operational within the university varies depending upon the point of view. The following sections describe the benefits to the individual users, the system administrators, and the university as a whole.

4.2.1 Provided to the Individual Users

Listed below are some of the benefits users may see as a result of a CFNS.

¹ Some of the "good causes" include marriage, divorce, and legally having a name change.

UNIVERSITY OF CALIFORNIA

Uniqueness

When a user first enters the CFNS system, they are given a unique login. In theory, they will have this account name for their entire association with the enterprise. This allows such things as backups and mail to be uniquely identified with a user across the enterprise. Since this information is keyed to user's association with the university, even years hence, each user can be identified uniquely and the user's files retrieved with a good level of certainty that they are being retrieved for the correct person.

Adulthood

Closely related to uniqueness (covered above), there is the benefit of adulthood that is imposed on users. When a user is given only one user id and is told that that id will be theirs for their entire university stay, any and all acts perpetrated by this user will be attached to this id also. Hence, doing malicious acts from the given id might be viewed by the more thoughtful user as "poor."

Interdepartmental data sharing

In the past if a user wanted to have the same data in two accounts, their only real options were along the lines of mailing the data between the two accounts. With a CFNS in place on all of the hosts involved, it is possible to automount the data across departmental boundaries within the enterprise. This is a direct result of having the UCI and NUID the same for each user on any CFNS administered host.

In fact, an enterprise wide file server might be employed to provide a centralized repository for all user data. Individual departments can then concentrate their computing facilities on providing computing platforms that are appropriate to that department's needs and leave the file backup and maintenance functions to a centralized service organization.

4.2.2 Provided to the System Administrator

In addition to the benefits that the user sees, a system administrator at a university also sees the benefits listed below as a result of using CFNS.

Start of semester crush

At the beginning of each semester, there is a large enrollment influx of new users, mostly students. To create all of these accounts by hand would be impossible. With a CFNS installed campus wide, adding an entire class of students to any machine across campus is as simple as adding a few lines to a configuration file, running a new enrollment database through the system, and installing the resulting password file. This entire process can take as little as 30 minutes.

Mail lists

In the past, different instructors have tried to keep track of which students are in their class for purposes of a class mail list. CFNS automates this procedure and makes sure that the list is correct, up to the last enrollment data feed.

4.2.3 Provided to the University

In the larger picture, a CFNS provides the following for all UNIX clients administered from one site:

- One login name and numeric user id for a user for their entire association with the enterprise.
- One account per user per machine, even if more than one class is taught on that machine. Therefore, there is no ambiguity in which class account maps to which user (there is only one account).
- Ability to authenticate backups for many years to come. Ie. ownership of the backups for the user "bob" can be identified as belonging to the person with OSU id number "111223333".
- No 'inactive users' on mailing lists as these are updated each time the password file is modified.

4.3 Costs

The development of UAMS, both on a site basis and later on the wider university basis, addressed many of the problems outlined in this thesis satisfactorily. However, these solutions did come at a cost, and as with any CFNS implementation "after the fact," some of these costs are quite large.

4.3.1 Preliminary Costs and Tasks

The establishment of a campus wide centralized flat name space system implies an incredible amount of work to arrange at this late date in the development of OSU's campus computing environment. One major task is the associating of an OSU id number with each and every user's account. Special numbers must be assigned to the mechanical accounts, such as "root" and "uucp," so that they can participate in much the same way as normal users.

Once all of the UCI's on all systems have an OSU id number associated with them, each user's account must be compared against all others across campus. This determines whether user has any other accounts across campus (as determined by the uniqueness of the OSU id number), as well as whether there is anyone else with the same UCI across campus (as determined by the required uniqueness of the UCI's). Those users who have the same UCI all across campus should have no change. Those users who have multiple login names that are not unique must be dealt with individually.

Since some users must have their login names changed to meet the new criteria, there can be some negative emotional response. Some users may even take offense at this entire process. A few speculative situations may arise as a result.

1. The user "bob" on one system whose name is changed to something else, say "bsmith," must unsubscribe to all mailing lists before the switch, then resubscribe with the new login name to avoid losing his mail. Analogously, if someone else has a login changed to "bob," the mail may get misdirected.
2. Professor "smith" comes back from vacation and is now "jsmith". After several failed attempts to login, the professor comes to the system administrator to ask what happened. After hearing the explanation, the professor is displeased with the change. In fact, professor smith has been putting email address on the net, in research papers, etc., as a source for some data/program/etc. that [s]he is providing to the net community. Now the linkage between

"smith" and professor smith (as "jsmith") must be reestablished. Also, the person who receives "smith" as a user id is inundated with requests for which there is no reasonable response.

3. Some user, "tom," knows the change is about to happen. Wanting to be offensive to the person who will be inheriting the "tom" account, this user posts something extremely offensive/stupid to USENET. After the change, the new person, possibly an upper level administrator, receives "tom" as a user id. Due to the propagation delays thru USENET, the new owner of "tom" now receives the replies to the postings made by the original "tom," for which there is no reasonable response.

It must be stressed that the better informed the user population is about both why these changes are happening and how they can benefit, the less reluctant it will be. Hence, the defusing action to be taken well before events get out of hand is to inform the user population of the benefits of this change.

4.3.2 Continuing

One of the continuing problems that UAMS must deal with on the OSU campus is that of OSU_ID changing. A bit of background is in order:

In the early 1980's, OSU started giving it's students photo id's with a printed OSU_ID number. These numbers typically looked like 000123456, and were created with the idea that one day they would be replaced by Social Security numbers. In fact, in 1986 or so, OSU made that switch and all students had to pay the bill of approximately \$10 for a new OSU photo id.

Unfortunately, this was mandated by the financial aid department, which said, in so many words, "to get financial aid on this campus, you must have your OSU_ID match your Social Security Number." For most of the natively born Americans, this was not a problem. However, for the international student body, this was a bit more involved.

The solution for OSU is to give an international student a photo id with one of the old style 000123456 OSU_ID numbers, since the administration *had* to be able to track the student, when they first arrived on campus. Then later, after the student had applied for a Social Security number,

change the student's OSU_ID number to the Social Security number. From that time forward, all enrollment data contains the new number.

The affect this had on UAMS was suddenly, the user with OSU_ID number 000123456 was no longer enrolled in anything. Hence that user's account, say "bob", was closed. At the same time a new user with OSU_ID number 111223333 was enrolled in some new classes. As the user name "bob" was already in use, a new one "bsmith" was created, along with a new NUID, password, etc.

The user would then come in and ask, "why doesn't my account work anymore?". After a bit of investigation, the two records for this person would have to be tracked down and merged, along with the data in their, now, two separate directories. The user would be reformed as to what the "bob" account's password was, then the merge would be finalized. Typically the next day, the user "bsmith" would be removed from the system and the user "bob" would be reattached to the files that used to belong to that user, completing the process.

Upon the installation of a CFNS, this "problem" should go away. With a centralized authority in charge of keeping track of the UCI and NUID when a user's OSU_ID changes, no UCI and no NUID change should occur since no "new" record is being created.

However, until the legacy accounts are merged into the CFNS, this merger process takes on a larger scope. Now, instead of one user's account on one system in one department, the capability exists for one user's account merger to affect literally dozens of individual accounts spanning the entire university.

4.4 Summary

This chapter has covered some of the costs and benefits associated with the change in paradigm to a Centralized Flat Namespace System.

Although most of the costs can have wide ranging consequences, they are for the most part one time costs. This fact makes these costs more bearable.

CHAPTER 5

BLACK BOX DESCRIPTION OF A CFNS

This chapter defines what a Centralized Flat Name Space system that would interface with UAMS would look like. Many of the terms used in this chapter have the same meaning as in the previously sections about UAMS.

5.1 Minimal Data Structures Needed

The CFNS needs a subset of the database fields that UAMS uses. Due to the operational usage of the CFNS (as outlined below), these structures are best split among several constituent databases. These databases would be joined (on the OSU_ID field) to form the appropriate fields for the given request. These databases are described below.

5.1.1 People Database

This database would be comprised of fields, each operationally described in Table 3.3, and presented in Table 5.1. The People Database would be joined to the others described in this section to form the overall view of the database. It would be maintained by that part of the registrars system that deals with name changes among students. These tend to occur as a result of marriages and divorces, but other causes do arise. This relation could also be used to maintain other data, such as preferred email addresses.

5.1.2 Association Database

The layout of this database is presented in Table 5.2. The Association Database would be the primary location for selection criteria in the creation of views of the CFNS database. The main mechanism for this is the use of the subfield code would indicate what kind of data subfield

osu_id	Student/Faculty identification number (KEY)
issue	Student/Faculty ID card issue number
fullname	Full name of the individual, as defined by the university

Table 5.1: People Database

osu_id	Student/Faculty identification number (KEY)
subfield code	A type for the data field below
subfield data	The content of type listed in the subfield code above

Table 5.2: Association Database

osu_id	Student/Faculty identification number (KEY)
subfield code	MAJOR
subfield data	A numeric major code

Table 5.3: Association Database, Example 1

data would contain. This allows a simple selection criteria to eliminate large numbers of the records contained in this database.

In Tables 5.3, 5.4 show two examples of how records in this database could look. In the first case, the record is identified as a major code record where the actual major code is contained in the subfield data would be, in this case, a four digit number. In the second case, the record is identified as an enrollment record where the actual class number (department prefix, course number, section, and type) would be contained in the subfield data.

This relation would be joined to the others to form the overall view of the database. This database would be linked into that part of the registrars system that deals with enrollment and college affiliation, among others.

5.1.3 Cross Reference Database

The final CFNS database would be the cross reference database. This would be used to form joins across the databases of the CFNS. It would be described as stated below, each field as described in Table 3.3.

This relation would be joined to the others to form the overall view of the database. This

osu_id	Student/Faculty identification number (KEY)
subfield code	ENROLLMENT, Spring of 1996
subfield data	A class that the given user is enrolled in

Table 5.4: Association Database, Example 2

osu_id	Student/Faculty identification number (KEY)
uci	Universal Computing Identifier (KEY)
nuid	Preferred Numeric User ID for NFS (KEY)

Table 5.5: Cross Reference Database

database would be linked into that part of the registrars system that deals with the instances of a student changing an OSU student/staff id number.

This relation is the key to the CFNS. It contains the crux of what the CFNS is designed to maintain. The contents of each of the individual fields within it is unique. That is for any one user recorded in this data field, their OSU_ID, their UCI and their NUID would be unique among all other entries in this database. This relation, along with the others previously mentioned in this section, duplicates the data content of the global record out of UAMS.

5.2 Supported Client/Server Functions

For the following discussion, a database remote procedure calls (RPC) will be the model for the interaction between the server and the clients in the CFNS system. This is in part driven by the need to secure the data that exists on the server from those who would be accessing the data. The RPC procedures would have access to the data. The RPC user would have access to the RPC, but not to the data. Hence the RPC user is insulated from the data by the view of the data the following RPC functions provide.

5.2.1 Make Return Set

This RPC will create an empty data set that will contain the keys into the CFNS that are selected by the Query By functions. This RPC has one argument, the Site Name that may be used by the Vote For Set command and the Return Set command.

5.2.2 Query By

This RPC will add CFNS keys to the return index data set created by Make Return Set. There are five different instances of this RPC, each having as an argument the search key as given by the

osu_id	Student/Faculty identification number
issue	Student/Faculty ID card issue number
fullname	Full name of the individual, as defined by the university
major	Student department affiliation
uci	Universal Computing Identifier
nuid	Preferred Numeric User ID for NFS
auto	Automatic enrollment rights

Table 5.6: Return Data Set Fields

instance. Namely, these five return types would be OSU_ID, NUID, UCI, class prefix, and MAJOR code.

5.2.3 Return Set

This RPC returns one record for each user who has a key specified in the return index data set, created by the `Make Return Set` RPC call and populated by the `Query By RPC` call(s). These records will contain the data outlined in Table 5.6.

5.2.4 Vote For Set

This RPC will add a vote record for the `Site Name` (that was specified in the `Make Return Set` RPC call) for each CFNS record that has a key specified in the return index data set created by the `Make Return Set` call.

5.2.5 Merge Ids

This RPC is used to handle the cases where a user is entered into the CFNS database with two different OSU id numbers. This RPC will have four arguments as show in Figure 5.7.

The first argument would be referred to as the “old” OSU id number. This is typically the original OSU id number that was assigned to the student prior to that student’s acquiring a Social Security number.

The second argument would be referred to as the “new” OSU id number. This is typically the Social Security number style OSU id number. For the purposes of the merger, this will be the resulting OSU id number for this user after the merger.

old_osu_id	Indicates the OSU student/staff id number of the record that is to be considered the older of the two to merge
new_osu_id	Indicates the OSU student/staff id number of the record that is to be considered the newer of the two to merge
uci_to_keep	This parameter indicates whether the uci in the record indicated by old_osu_id or the uci in the record indicated by new_osu_id should be kept in the new record
nuid_to_keep	This parameter indicates whether the nuid in the record indicated by old_osu_id or the nuid in the record indicated by new_osu_id should be kept in the new record

Table 5.7: Merge RPC Parameters

The third argument would say that in the resulting merger of the “old” and the “new” OSU id number records, either the “old” record’s UCI or the “new” record’s UCI would be retained.

The fourth argument would say that in the resulting merger of the “old” and the “new” OSU id number records, either the “old” record’s NUID or the “new” record’s NUID would be retained.

Experience with UAMS has shown that one of the most troubling aspects of keeping the database synchronized with the real world is maintaining the veracity of the OSU_ID numbers. This is in part due to the fact that due to certain restrictions placed upon the keeping of student records for financial records, their OSU_ID numbers are preferred to be their Social Security Numbers. However, due to the way the enrollment system works, when a user changes their OSU_ID number, the only indication is that one user, with an old style id number, drops all enrollment and another with a similar (but not necessarily the same) name enrolls in some (but not necessarily all) of the same classes.

This RPC is a concession to the reality that the CFNS database will most likely never be in complete synchronization. This RPC will allow the various site administrators to do some database auditing.

5.2.6 Create A New User

This RPC allows sites other than the central enterprise administration to create new users, previously unknown to the system. It takes as arguments an arbitrary OSU_ID number¹, a text string that will

¹ Typically these users will have a “+” record as described in Section 3.5.3.

be considered the FULLNAME field, a Suggested UCI, and a Suggested NUID. It then would follow the general UCI creation heuristic as outlined in Figure 3.4. Lastly, it would return the generated data as outlined in Table 5.6.

5.3 How it all Fits Together

The key concepts presented in Figure 3.3 and Section 3.13 are what the previously outlined RPCs would replace. These messages, passed between the CFNS database server and the departmental "slave" UAMS servers, would provide the individual departments that currently participate in the UAMS system the same basic services. The only real change to Figure 3.3 would be to eliminate the time delays between events. This would allow faster transaction processing time².

Once the CFNS database and server were in place and operational, the slave UAMS database servers could be migrated away from the current master UAMS server onto the CFNS database server. Further, as the individual departments would still have the functionality of their slave UAMS servers, their day to day operations should not be impacted. Hence all of the services that these slave UAMS servers supply to their departments should not change.

In addition, as the constituent databases outlined in Section 5.1, once joined by the various RPCs outlined in Section 5.2, would eliminate the need for the master UAMS server's database. In fact, the entire reason for the master UAMS server would be eliminated with the creation of a CFNS. Thus the master UAMS server could be halted.

5.4 Other Issues

There are several other issues that will no doubt come to light in the process of trying to implement a CFNS. Some of those foreseen are covered in the the following sections.

5.4.1 Data Base Inconsistencies

As has been mentioned, synchronization of the CFNS that UAMS has maintained with the data from the registrar has been difficult. With the advent of a CFNS supported and maintained as part

² Although it would allow it, the CFNS would not necessarily force it.

of the central campus computer enrollment system, notification of changes in data can be provided to the CFNS automatically. Hence, the entire scenario given in Section 4.3.2 can be avoided.

5.4.2 Purging the Data Base of Old Records

The database that UAMS currently maintains contains many cases of "duplicate ids". This is a result of the OSU id changing problem outlined above. In the final version of the CFNS database, these users would have their records merged under their current OSU_ID number to consolidate the logins and to assure the "one user, one login" premise of the CFNS. This will be simplified once the CFNS is in place as the central university administrative database has the mapping of which old style OSU_ID maps to which user as part of their historical database. This mapping, here to unavailable to UAMS, will make the merger of these two records technically a fairly easy procedure.

In the practical implementation of this purging, some of the same problems addressed in Section 4.3.1 may occur. In addition, the merging of files from these two distinct user accounts may prove quite difficult.

5.5 Summary

In talking about a CFNS, this chapter has shown how a CFNS will face many of the same problems currently faced by UAMS. Further, a CFNS properly interfaced to the central campus administrative databases would be able to deal with some of these problems in a far more automated way than currently available to UAMS. These concepts suggest that a CFNS would allow a performance beyond what is currently available from UAMS while not sacrificing those services currently available to the individual participating departments.

CHAPTER 6

PROPOSED FUTURE WORK AND CONCLUSIONS

6.1 Future Work

The ultimate form this code may take is hard to gauge at this point. This section presents a few of the directions the UAMS project could take. This is of course, not a complete list.

6.1.1 UAMS-lite

Given that empirical tests of speed prove satisfactory, one thing that could be done with UAMS would be to move the entire database onto a central host and use a commercial database and a commercial access system to grant access to the data remotely. Then the only blocks of code existing in the UAMS-lite would be those involved with processing these data streams into the correct format for a given client.

This approach would allow all of the current "manual" maintenance of the relational database that is UAMS to be offloaded onto the commercial database engine. Further, as the database would then truly be a relational database, the time currently spent maintaining the inverted indices into the DBM based database would be eliminated.

This approach would violate several of the original design goals of the original UDB. It would involve the use of a commercial database engine and a, presumably, commercial access method, both of which would be purchased. Further, the use of these commercial products might limit which kinds of platforms could then be UAMS-lite servers.

6.1.2 Real Time Access to the Master UAMS Site

One of the original design goals of the original UDB was to avoid using such things as TCP/IP networking. At the time, there were several systems within the Computer Science Department that simply did not have TCP/IP networking interfaces. This caused UDB, and later UAMS, servers to rely on electronic mail interfaces as the data transport method between servers.

Currently however, all of the UAMS server sites have TCP/IP. Hence, server to server commu-

nications could be moved from the currently slow email batch processing system to a more direct group of peers topography.

6.2 Conclusions

The current state of this project affords the system administrator an extremely useful set of tools for managing the user accounts across a heterogeneous computer network. These tools will continue to be used here at OSU for many years to come and should be easily modified to support any new operating system that the Computer Science Department might choose to run.

BIBLIOGRAPHY

- [BGMR94] Andrew Bettison, Andrew Gollan, Chris Maltby, and Neil Russell. Share ii - a user administration and resource control system for unix. In *Usenix Conference Proceedings, Large Installation Systems Administration V* [USE94], pages 51–60.
- [CKCS90] David A. Curry, Samuel D. Kimery, Kent C. De La Croix, and Jeffrey R. Schwab. Acmaint: An account creation and maintenance system for distributed unix systems. In *Usenix Conference Proceedings, Large Installation Systems Administration IV* [USE90], pages 1–9.
- [DLM90] William A. Doster, Yew-Hong Leong, and Stephen J. Mattson. Uniqname overview. In *Usenix Conference Proceedings, Large Installation Systems Administration IV* [USE90], pages 27–35.
- [EVS88] Mark E. Epstein, Curt Vandetta, and John Sechrest. Asmodeus - a daemon servant for the system administrator. In *USENIX Conference Proceedings, Summer USENIX '88*, pages 377–391. USENIX Organization, June 20-24, 1988.
- [Fin92] Jon Finke. Automated userid management - simon management system. In *Community Workshop*, pages Subject Area 3, Paper 5. Rensselaer Polytechnic Institute, 1992.
- [Har94] M. Harlander. Central system administration in a heterogeneous unix environment: Genuadmin. In *Usenix Conference Proceedings, Large Installation Systems Administration V* [USE94], pages 1–8.
- [III84] Douglas P. Kingston III. Mmdfii: A technical review. In *Usenix Conference Proceedings, Summer Conference, Salt Lake City*, pages 32–41. USENIX Organization, 1984.

- [Inf91] Information Builders, Inc. *EDA API/SQL and EDA/Structured Query Language Reference Manual*, 1991.
- [Sto93] Roland J. Stolfa. Simplifying system administration tasks: The uams approach. In *Usenix Conference Proceedings, Large Installation Systems Administration VII*, pages 203–208, Monterey, California, USA, November 1-5 1993. USENIX Organization.
- [Sto96] Roland J. Stolfa. Uams: The man pages. Technical Report OSU-CS-TR-96-2, Oklahoma State University, Computer Science Department, 219 Math Sciences Building, Stillwater OK 74078, May 1996.
- [SV90a] Stephen P. Schaefer and Satyanarayana R. Vemulakonda. newu: Multi-host user setup. In *Usenix Conference Proceedings, Large Installation Systems Administration IV* [USE90], pages 23–26.
- [SV90b] Roland J. Stolfa and Mark J. Vasoll. Udb - user data base system. In *Usenix Conference Proceedings, Large Installation Systems Administration IV* [USE90], pages 11–15.
- [Urb90] Michael Urban. Gaud: Rand's group and user database. In *Usenix Conference Proceedings, Large Installation Systems Administration IV* [USE90], pages 17–22.
- [USE90] USENIX Organization. *Usenix Conference Proceedings, Large Installation Systems Administration IV*, Colorado Springs, Colorado, USA, October 17-19 1990.
- [USE94] USENIX Organization. *Usenix Conference Proceedings, Large Installation Systems Administration V*, San Diego, California, USA, 1994.

APPENDIX A

GLOSSARY

CFNS Centralized Flat Namespace System. Account management paradigm where each physical user is granted one and only one login name and one and only one numeric user id number.

Login Name A text string that is used as a mnemonic for the computer to distinguish one user from another. Login Names are typically converted to a NUID by the operating system improve performance. See also PUD and UCI.

NFS The Network File System. Used to allow heterogeneous computer systems to share files across a computer network.

NUID Numeric User Identification. Used by operating systems to internally differentiate users.

Password An access control mechanism common among commercial operating systems to secure the data belonging to any user.

POP Post Office Protocol. A method of storing and retrieving electronic mail from a centralized spooling/storage site.

PUD Personalized User Identifier. A term used by the Computing and Information Services group on campus to describe personally chosen login names. Synonymous to UCI.

RAF Rights Alias File. Central configuration mechanism on a server site that defines which URI's affect the transfer of a user's data from the server to a given client. See Section 3.3.

RPC Remote Procedure Call. A set of routines that allow application programs to execute procedure calls across the network. Destination machines may be heterogeneous[Inf91].

UAMS The User Account Management System. Was the successor to UDB.

UCI Universal Computing Identifier. Synonymous to login name for most purposes. See Section 3.5.

UDB The User Data Base system. Existed as a distinct software package from 1987 through 1990.
Predecessor to UAMS.

URI Universal Right Identifier is a text string given to an UDB/UAMS record that associated that record with some group of other records. Examples include class enrollment rights, major codes, etc.

2

VITA

Roland Joseph Stolfa

Candidate for the Degree of

Master of Science

Thesis: UAMS, A STUDY IN SYSTEM ADMINISTRATION AUTOMATION

Major Field: Computer Science

Biographical Data:

Personal Data: Born in Ardmore Oklahoma, on December 20, 1963, the son of
of James and Pauline Stolfa.

Education: Graduated from Ardmore High School in 1982;
attended St. Gregory's College 1982-1984; graduated from Oklahoma State
University, Stillwater Oklahoma in 1986 with a Bachelors of Science in Computer
Science., Completed the requirements for the Master of Science degree with
a major in Computer Science at Oklahoma State University, May 1997.

Experience: System Administrator, Department of Electrical and Computer Engineering,
Oklahoma State University, August, 1985 to May 1987. Data Communication
Specialist, University Computer Center, Oklahoma State University, January 1988
to July 1988. System Administrator, Computer Science Department, Oklahoma
State University, July 1988 to present.