The University of Kansas

Information and Telecommunication Technology Center

Technical Report

ACE Architectural Design Version 1.0

Renzo Hayashi, Leon Searl, and Gary Minden

ITTC-FY2002-TR-23150-02

December 2001

Project Sponsor:
U.S. Air Force and the Defense Advanced Research
Projects Agency under contract no. F30602-00-2-0581
and The National Science Foundation
under grant EIA-9972843

Table of Contents

1.0 Introduction

The ACE structure is an intricate system of services all interconnected via a large computer network to provide a wide range of capabilities to its clients and users. These services must coexist and work together seamlessly to provide a long-lived and robust environment where users can access computational resources on demand at the touch of a button. Refer to the Ambient Computational Environments proposal document for details on the ACE concept.

In order for these services to provide such invisible integration of capabilities within a highly complex system it becomes necessary to develop a very modular and flexible service architecture that can fit the requirements of an ACE system (see ACESoftwareRequirements document for further details).

The purpose of this document is to outline and describe in detail the top-level view of the entire ACE service architecture. This document covers the basic infrastructure of the ACE system, the types of services it can and should provide, and how all these services interact to accomplish their tasks and serve ACE users. This document is intended to be an all encompassing bird's eye view description of ACE and how it works. Further details on the elements described within this document may be obtained by referring to their appropriate specifications and design documents.

2.0 ACE Architectural Design

2.1 Overview

The ACE infrastructure is divided into many unique application and design areas, all of which shall be discussed in this document. These main areas are as follows: ACE users and human interaction with widespread service accessibility (including user work contexts); ACE security and authentication; ACE infrastructure and service daemons; basic ACE framework services, ACE user applications; and persistent store.

All these parts are essential to the development of a secure, long-lived, and widely accessible computational resources available throughout the ACE environment. Following, these concepts and the details on these components and how they interact shall be further developed.

2.2 ACE Users & Human Interaction

The single most important reason for the existence of an ACE system is that of serving an ACE user. ACE users are the end consumers of all services provided by an ACE. Without them there would be no need for an ACE.

This section shall briefly discuss what it means to be an ACE user, how users can interact with an ACE, and the concept of user workspaces and how these are dealt with within ACE.

2.2.1 Who are ACE Users?

ACE users can be separated into two main categories: human users and non-human users. Non-human users are high-level applications that utilize ACE services on their own to provide automation within an ACE. Examples of this would be video monitoring systems, personnel tracking systems, etc…

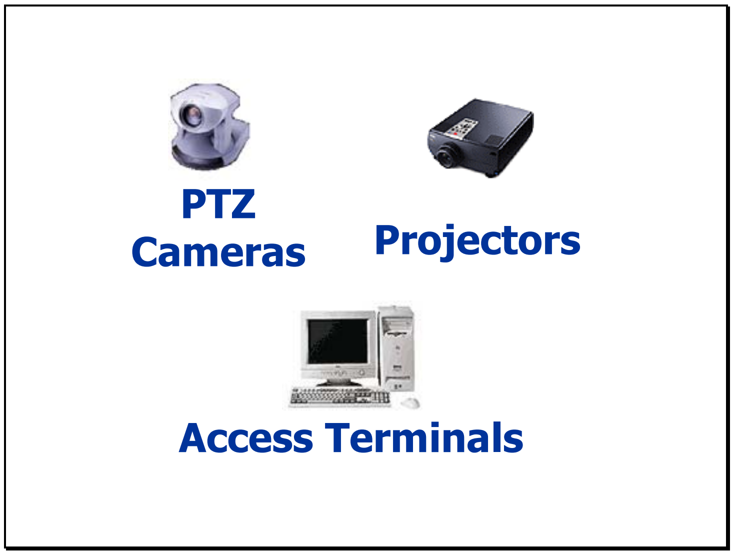
The bulk of ACE system usage shall come from human users. This shall be the focus of the remaining part of this section and most of this document. Furthermore, from now on, ACE users shall strictly refer to human users. As part of the general ACE requirements, users shall be registered with a central ACE database (for a single ACE) and each shall be uniquely identified via their public key.

Human users are the ones who shall be accessing an ACE environment given proper identification and calling on ACE services to command devices and utilize the ACE distributed computing power.

2.2.2 Human Interaction

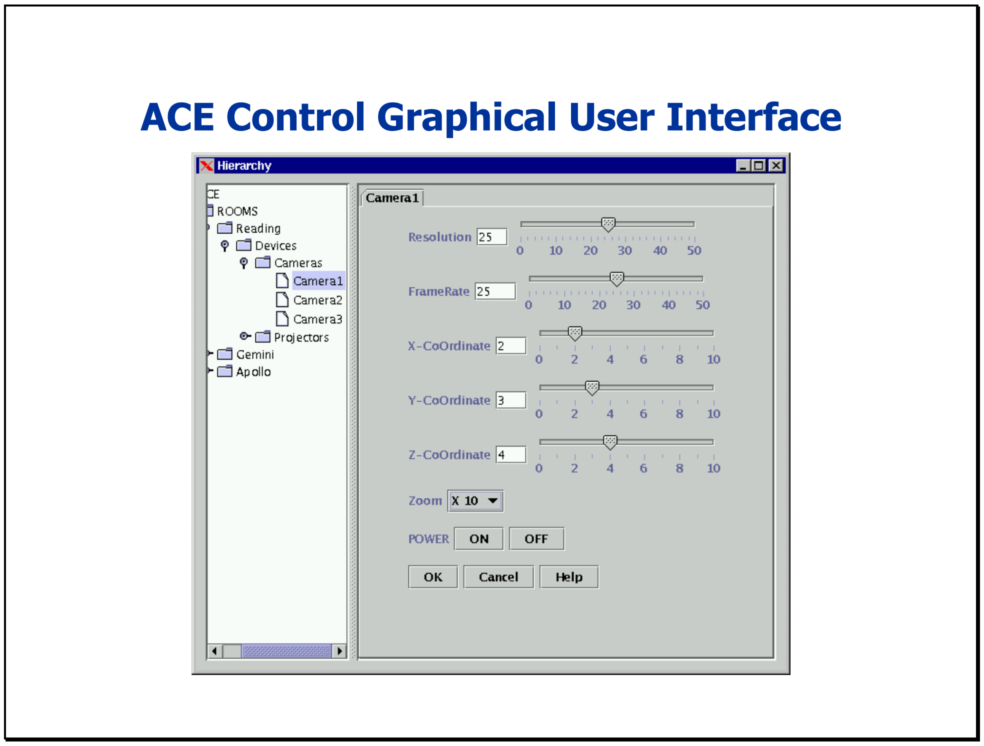
ACE users can utilize an ACE in various and unlimited ways. It all depends on the services that are defined within a specific ACE and how these are interconnected to provide specific functionality.

ACE users can interact with an ACE through various different ACE enabled devices. In order for a device to be ACE enabled, it must have low-level interface software developed for it so that it may be integrated into the ACE service infrastructure. Such devices include PTZ (Pan-Tilt-Zoom) cameras, projectors, identification devices such as fingerprint scanners and iButton readers, etc. Other devices such as notebooks, picturebooks, PDAs (Personal Digital Assistants), and workstations simply serve as networked access points to the ACE network. These devices shall be distributed throughout an ACE within conference rooms, offices, hallways, and even automobiles.



Some ACE User Interaction Devices

User interaction isn't limited to the devices mentioned above. Many forms of interaction with an ACE shall come from high-level programs and GUIs (Graphical User Interfaces) intended to accept ACE user commands, interpret these commands, and call on other ACE services to execute them.



Resolution 25
FrameRate 25
X-CoOrdinate 2
Y-CoOrdinate 3
Z-CoOrdinate 4
Zoom X 10

The picture of the graphical user interface above shows an example of how users may access an ACE and control devices and services available. On the left side, available ACE services and devices are listed in a hierarchical tree fashion based on their location within ACE (e.g. specific rooms in a building). By selecting a service or device on the left side, the appropriate parameter controls are displayed to the right. The example seen in the GUI above is for the PTZ camera. Notice that the right side allows the user to control parameters such as x, y, and z coordinates of where the camera is pointed at, the resolution and frame rate of image capturing, zoom factor, and also provides an on/off button to switch the camera on and off.

Examples of other user interaction software are voice recognition systems, voice command interpreters, computer speech synthesis systems, device control graphical interfaces, face and gesture recognition systems, sound triangulation systems, user locators and trackers, audio and video streaming, telephone over IP systems, etc.

All these devices and software services are integrated into the ACE architecture and infrastructure (which shall be further discussed below) and thus can easily communicate with each other and form a seamless network of services.

Other ACE User Interaction Devices

The ACE Software Requirements document has more details on specific requirements for ACE and its capabilities.

2.2.3 User Workspaces

Another significant form of access that an ACE provides for its users is with user workspaces. An ACE user workspace is a virtual computational space/environment that a user may use to run his/her applications and access the network. This virtual space is defined physically as the graphical space on a terminal or workstation screen where a user's file space can be accessed, applications and programs can be executed, and computational power utilized.

A user may have more than one instance of a workspace defined for him/her. In each workspace the user may run different programs while accessing the same user file space.

In an ACE, a user may identify him/herself via an ACE identification device in order to have his/her active workspace brought up on demand independent of where the user is within the environment. Through his/her active workspace, the user may bring up necessary files, and run desired applications. Then, upon leaving or logging off from a certain access location within an ACE, the workspace and its current state are maintained. The user can then pick up where he/she left off at another access point at a later time.

In a sense, a workspace is a virtual version of a user's personal desktop computer. The difference is that it has no physical dimensions. Therefore it may be taken immediately to an access location where the user is and utilized at the push of a button.

2.3 ACE Security & Authentication

In this section, ACE security measures shall be discussed. These are necessary to make ACE a network where only valid users with proper permissions may run applications, access files, and transmit information in a secure fashion.

2.3.1 ACE Communications

As with any network environment, the communication framework must provide a means of relaying information from point A to point B in a secure manner. Furthermore, this communication must be properly verified and allowed to occur.

These security and authentication mechanisms have also been incorporated into the ACE service infrastructure in order to make sure that services are accessed by authorized users and/or programs, and that the communication between them is secure from eavesdroppers and unauthorized entities.

All ACE communications from one service to another is encrypted using SSL (Secure Socket Layer) level encryption. All the service commands using the ACE command language (more details later) and all data transferred between ACE services are encrypted in this fashion at the socket layer.

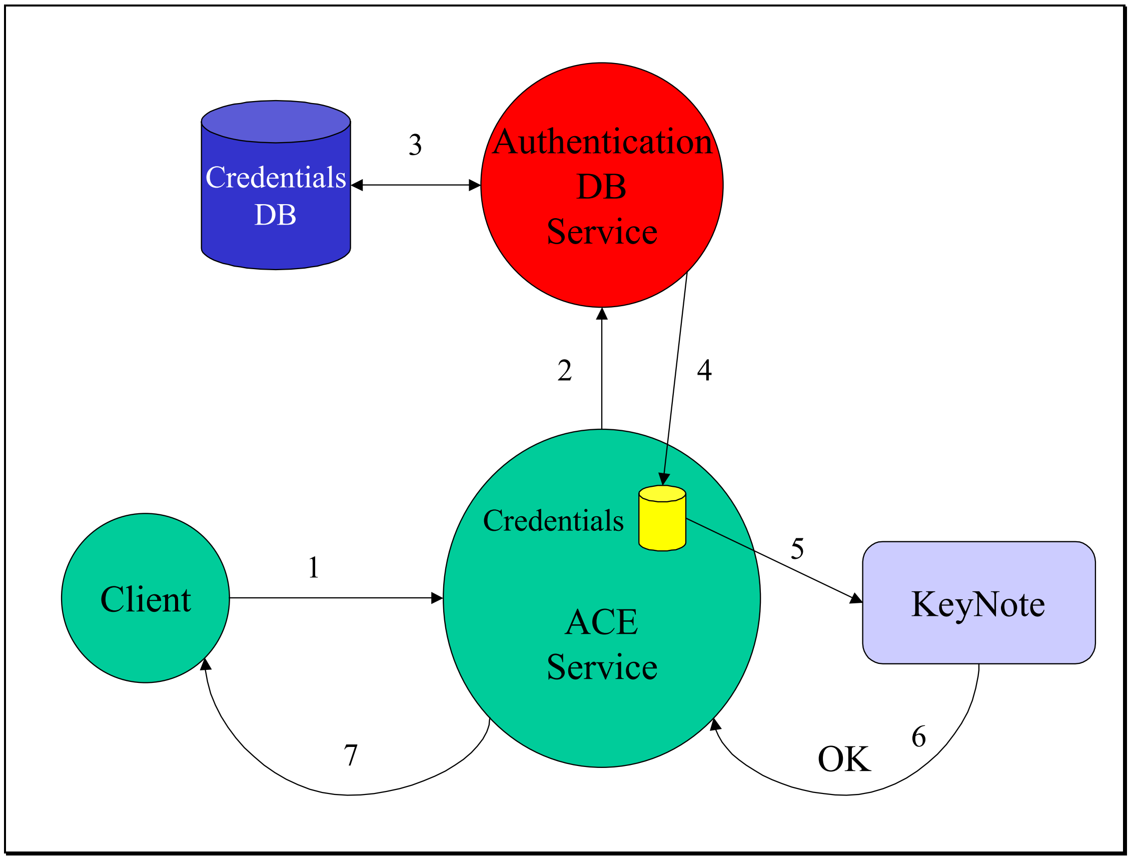
Security requirements details can be viewed in the ACE Security Requirements document.

2.3.2 ACE Service Access & Authorization

Within an ACE network ACE users must be given access to underlying network and also be registered with ACE in order to access and utilize ACE services. In order to be properly identified within an ACE the user must first access an ACE user identification device such as a fingerprint scanner, an iButton reader, or a badge reader. Once this is done and the user has been identified, he/she may access the services within an ACE. Unfortunately, being a valid ACE user does not get a user far if he/she doesn't have valid permissions.

In ACE, both users and services are restricted in the use of services and what commands may be issued. For this purpose, the KeyNote trust management system has been integrated into the ACE service infrastructure. Both users and services shall have credentials and assertions defined for what can and can't be done within an ACE. These credentials control what commands can be issued, what services can be accessed, what connections can be made, for how long services can be utilized, how much of computing resources may be consumed, etc.

Below is an example of how ACE services verify permissions for a simple ACE service:



In this example, a client wishes to execute a command on an ACE service (1). The ACE service recognizes that a command is about to be executed by a specific client and goes on to request the available KeyNote credentials and assertions for this command and for this specific client from the Authentication Database Service (2). This service serves as an interface to the storage and access of credentials and assertions to other ACE services wishing to verify client trust. The Authentication DB service looks up the necessary information from the database (3) and returns the requested information to the ACE service (4). The ACE service then forwards this information to the KeyNote trust management system (5), which verifies that the provided credentials and assertions are valid for this type of client and action. Then either an OK or a NOT OK is replied back to the ACE service as to the validity of the client's permissions. In this case, the OK is given (6) and the ACE service performs the command requested by the client and returns the result of the executed command to the client (7). Had a NOT OK been given, the ACE service would refuse to execute the requested command replying that the client did not have valid permissions for the attempted command.

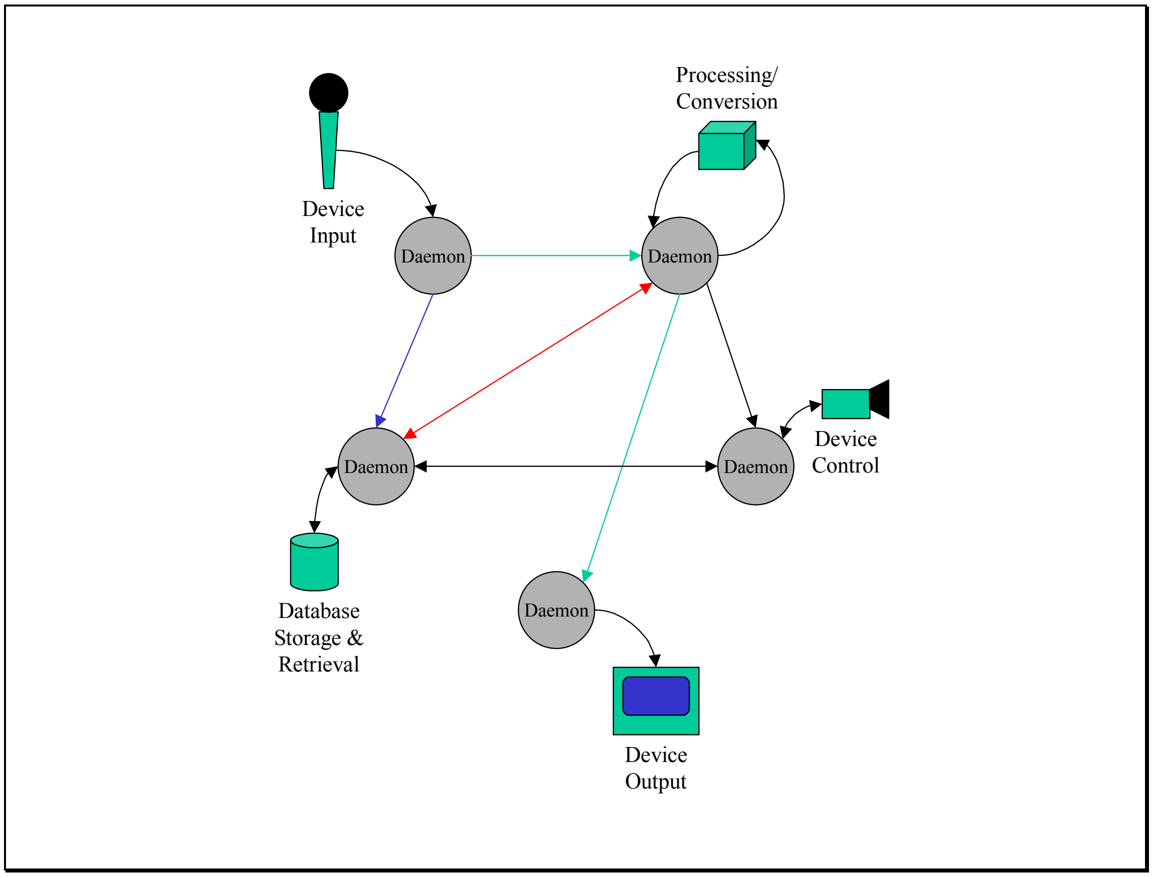
For further information on authentication requirements for ACE, refer to the ACE Authorization Requirements document and for details on KeyNote usage in ACE see the ACE Project KeyNote Usage Requirements document.

2.4 Daemon Infrastructure Design

What exactly is an ACE service? An ACE service is any type of functionality that can be provided to an ACE user or another ACE service within the ACE network. Whether this be data processing, data input, data storage, data transmission, device control, network access, etc, a service provides users such capabilities. In order to provide ACE users with a wide range of distributed services and as part of the ACE requirements of modularity and flexibility, it became obvious that ACE in itself would not be a single all knowing omnipotent computational entity that controlled all sub-environments within a single ACE. In fact, the concept of ACE required much the opposite style of computing where small and independent systems work together to endow ACE with its unique capabilities. Simple systems that can come together like building blocks or lego pieces. These pieces, when in unison, can produce complex output and/or behavior.

This reasoning led to the design of a basic ACE service daemon that is responsible for performing a single, unique, and well defined function within the ACE infrastructure. Thus, different and specialized daemons comprise the building blocks of the ACE world. These are the independent pieces that work together to provide a variety of more complex services/capabilities within ACE.

These unique services/daemons act as independent entities with unique capabilities and can serve as both consumers or producers of services to any other service. They must communicate with each other in a common language that can be easily parsed, understood, and can account for all commands and data transferred in between them. The diagram below depicts this concept:



As was stated before, the arrows connecting these services represent a singular and well-defined language understood by all service daemons.

Specifics on ACE daemon requirements can be found in the ACE Daemon Requirements document.

2.4.1 Daemon Hierarchy

Another significant aspect of the ACE daemon infrastructure is its modularity and ease of integration into the existing architecture.

The basic service daemon is primarily composed of three modules: the daemon itself (implementation), the service semantics, and the command interface to that daemon. Each daemon command interface provides the means by which users and other services access and command service daemons through the network.

The service daemon itself is what executes the commands it receives, operating on data, performing calculations, or serving as yet another interface to something like a database or GUI.

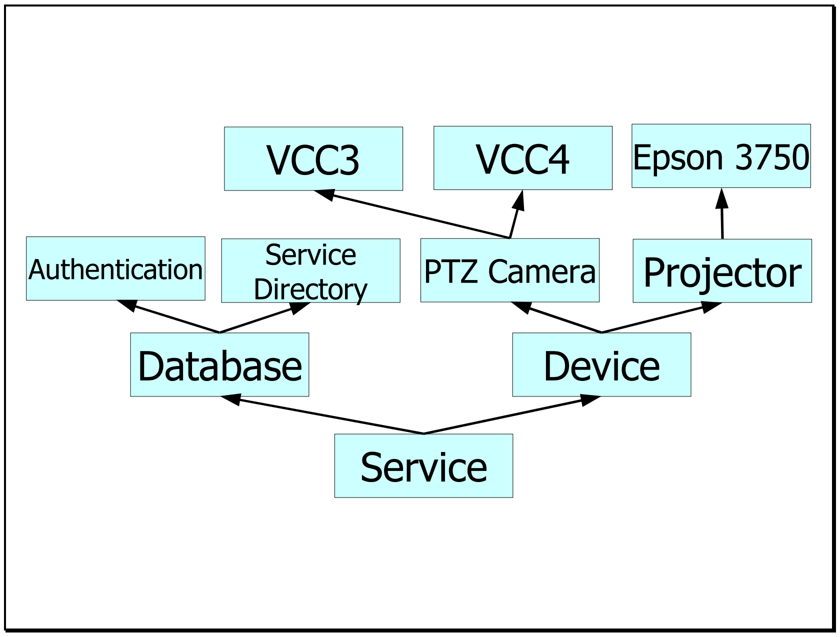
The service semantics define that valid commands and arguments to those commands that are valid and understandable by the service daemon. These semantics are used in the construction and parsing of service commands that are transmitted from one daemon or client to another.

Finally, the command interface is what provides users and client programs the means by which to execute or request services from service daemons. These interfaces are arranged in a hierarchical manner. In this way, child nodes inherit methods, characteristics, and actions from the parent nodes. With this structure, it is easy to develop different services and integrate them into the current infrastructure. Additionally, with this type of inheritance structure, child nodes can be developed to be like their parent nodes but with a few more functionalities.

As can be seen from the diagram below, all services inherit from a basic service daemon. These are then divided into more types of services (only database and device are illustrated here). These in turn define other child database and device control services. Finally, the device control services can be further specialized for specific models of PTZ cameras and projectors.

With this infrastructure, new services can be brought online quickly and painlessly. Device and service specific portions of new services to be added are isolated to prevent complete re-coding of service daemon specific framework.

Another advantage of this modular daemon structure is that the underlying details of the service implementation are completely hidden from the client and its programmer. All that needs to be known is the command interface to the service and how to utilize it to obtain the capabilities needed.



2.4.2 Command Language & Parser

As previously stated, the services within an ACE share a common control language. This language is used by the ACE daemons to communicate both data and service commands.

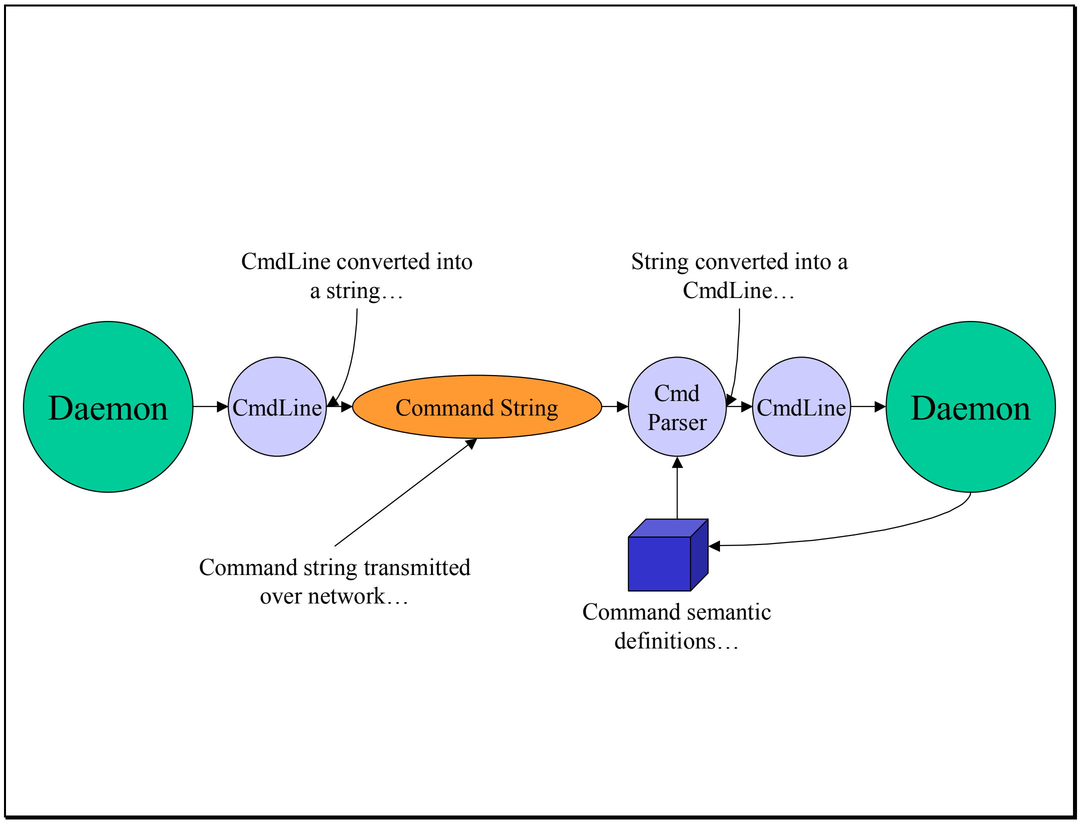
The structure of this command language is based on the simple command and arguments structure much like can be seen in a regular Unix environment.

<command> [<argument>…];

For details on the ACE command language syntax refer to the ACE Language Specifications document.

For each unique daemon implementation, a set of unique command and argument semantics must be defined, within the basic language structure, and tailored to fit the specific capabilities of that service daemon. That is, for each daemon that is implemented for a specific service there must also have been defined for it the commands that it'll understand, execute, and return (return commands are also used to reply on the status of the attempted command – successful or failed) and the arguments that go along with those commands.

All this syntactic and semantic content is stored within a basic ACE service command data structure used by clients and daemons. This object that is used for storing and communicating ACE commands is called the `ACECmdLine` object. See the ACECmdLine specifications document for further details. Every command that is to be issued to an ACE service is first built as an `ACECmdLine` object. This object is then converted into a string by the issuing client/daemon and is then transmitted over the network to the receiving side. Once received, this string is used to construct an exact copy of the `ACECmdLine` object by means of the ACE Command Parser. The parser checks the incoming string for syntactic and semantic correctness (against those parameters defined for the receiving daemon/service) and the new `ACECmdLine` object is constructed. Once this is done, the recipient may access the transmitted content of the command in order to perform its task. See the diagram below for a depiction of this process.


CmdLine converted into a string…
String converted into a CmdLine…
Daemon
CmdLine
Command String
Cmd Parser
CmdLine
Daemon
Command string transmitted over network…
Command semantic definitions…

2.4.3 The ACE Service Daemon

Each service and/or device within an ACE is controlled directly by a unique instance of an ACE service daemon. This daemon is responsible for being an intermediary between the service/device control platform and the client wishing to utilize the service/device. Note here that when a reference is made to a "client" within an ACE that may refer to any kind of entity attempting to communicate and/or request services from an ACE daemon. Such an entity may be a simple client program or yet another ACE daemon. Design details on ACE inter-daemon communications can be found in the ACE Connection Interface Design document.

The daemon provides a structure for encrypted and certified socket communications, service registration, and lease renewal (ASD discussed below). With this framework in place, implementing a service daemon and along with all its necessary functions and command semantic definitions (based on the ACE command language) becomes a simple, standard, and modular task. As services are implemented these can then be made to talk to one another and provide services to one another. Thus building a complex system of ACE services becomes a matter of simply defining functions and proper interface commands between such services.

Furthermore, being built as independent threads of execution, daemons may be run on separate machines spread throughout an ACE network. Also, each machine/computing system in an ACE may have running within it one or more ACE service daemons, each providing a specific service.

For more details on the inner workings and functions of the ACE Daemon and it's components refer to the ACE Daemon Design document and the ACE General Service Client Daemon Design document.

2.4.4 Service Discovery – ACE Service Directory

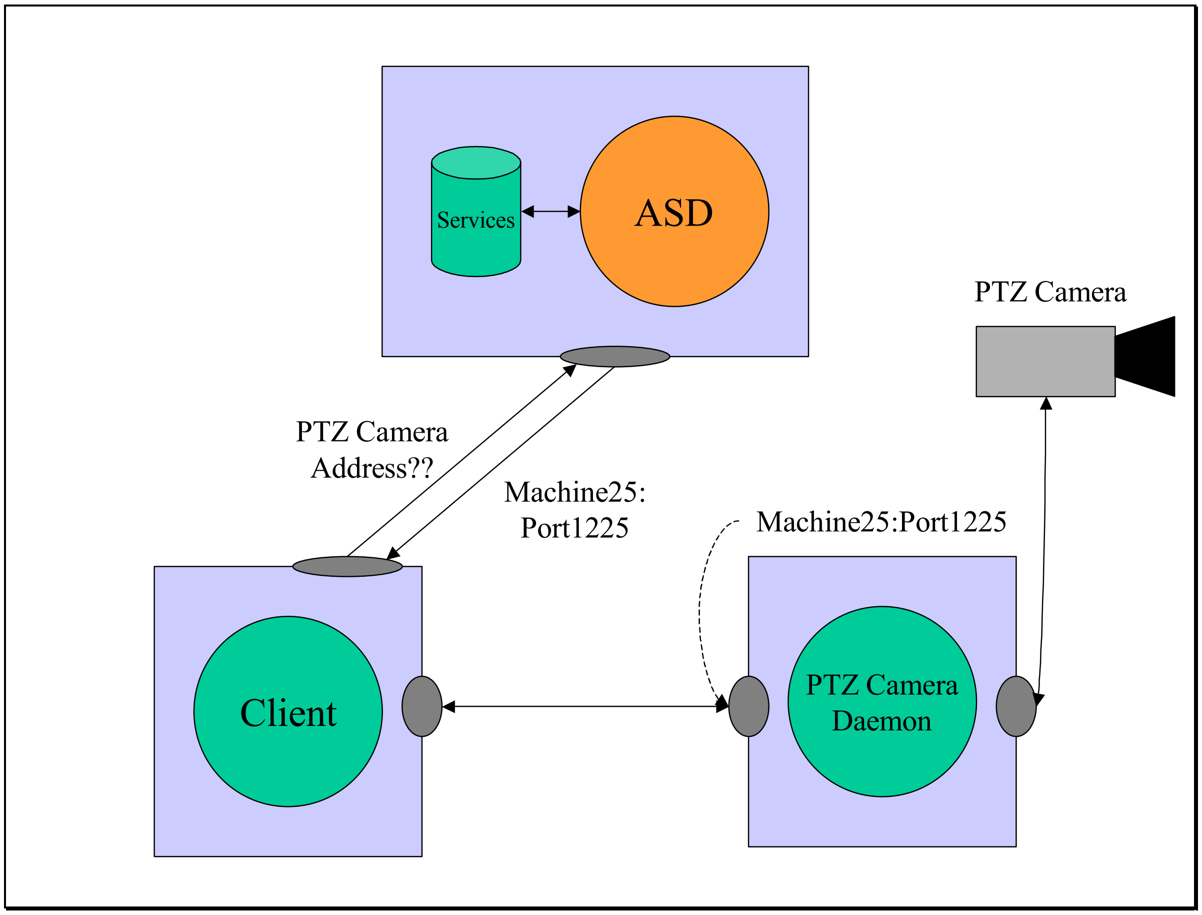
If so many different unique services can be made available within the ACE infrastructure, how does one service find another? With services distributed throughout the environment it becomes necessary to provide other services and clients with the ability to quickly find and connect to desired daemons. ACE needs to provide service with this capability. A capability similar to a DNS in which a central location or server can be queried for finding specific machines within a network. ACE needs to supply services with the proper information for finding other services within the ACE network.

Within this daemon infrastructure it becomes sensible to make such lookup and discovery ability yet another service provided by an ACE daemon. This service is called the ACE Service Directory – ASD. This service shall serve as a central listing or directory of services currently available and running within the ACE environment. Any and all services currently active must register themselves with the ASD service (the location of which is known to all ACE daemons) so that other services know that it is active and available.

Also, it is necessary for these services to remove themselves from the ASD registry upon shutdown by properly informing the ASD service of their removal.

Furthermore, if services become inactive due to programming errors or system crashes, it is necessary for the ASD to remove those services from its listing so that other services don't waste time and resources attempting to connect to a defunct ACE service. A simple mechanism to do so is to implement service leases within the ASD. Upon registration with the ASD, each ACE service is given a lease time for which they'll be allowed to remain within the ASD listing. If a registered service fails to renew its service lease with the ASD upon lease time expiration, this service shall automatically be removed from the ASD. In order to remain in the ASD, services must periodically issue service lease renewals with the ASD so that they may remain registered within it. This mechanism accounts for the system failures mentioned earlier whereby daemons that become inactive due to malfunction are automatically removed from the ASD once their service lease expires.

The picture below demonstrates how the ASD is used.



Here a client program wishes to communicate with a PTZ Camera service. By knowing the fixed location of the ASD, the program inquires if there are any PTZ camera services available and if so, where. The ASD looks up its database of currently active services and finds a PTZ camera service available and responds with the machine and port address of that service. Once the reply is received, the client program opens a connection to the service at the address specified by the ASD and begins controlling the camera at that location via commands to the PTZ camera daemon.

See the ASD Design document and the ASD Database Design document for more details on the ASD and the information it maintains.

2.4.5 Daemon Notifications

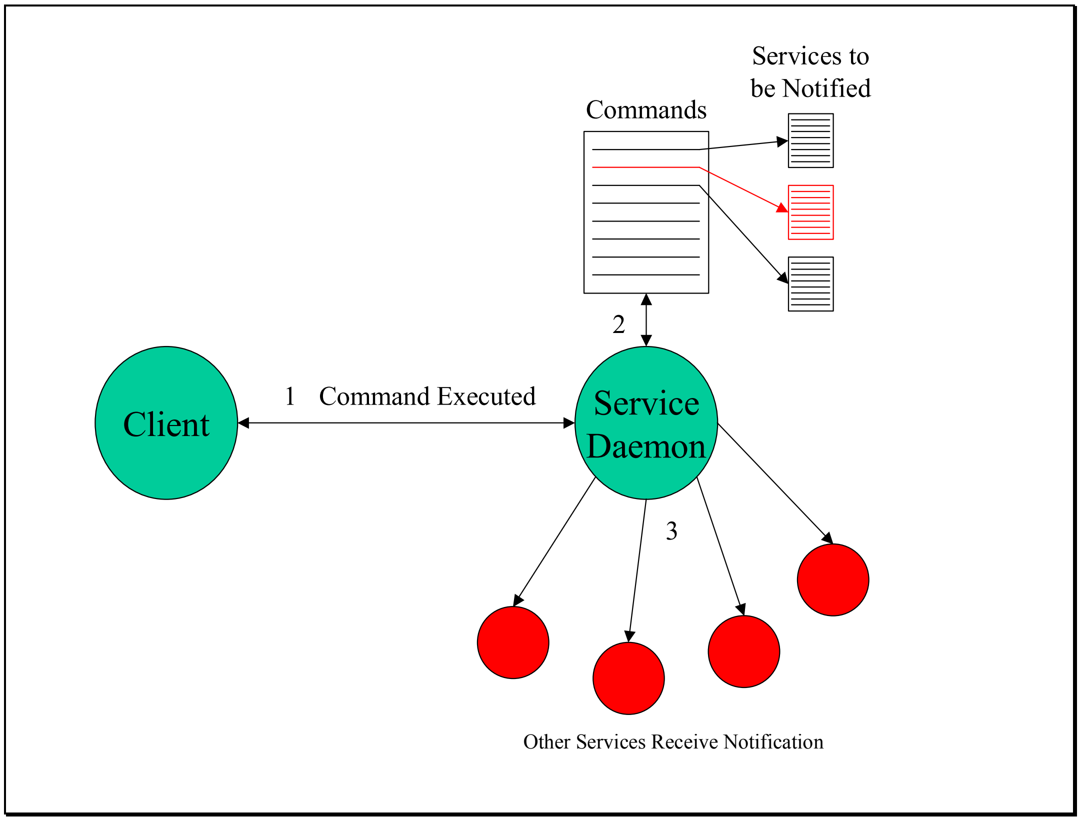
Another important component of an ACE service are notification commands. The basic ACE daemon has incorporated within it the capability of notifying interested parties (other services) of specific commands that have been requested and that it has executed.

This capability is what is called "ACE service notifications" and it is inherent in all ACE daemons within the Service level (see section 2.4.1). Simply put, all ACE daemons have notification commands semantically and syntactically defined for them (ACE command language). These commands allow ACE services to keep a running list of all other ACE commands that are being "listened" to and all the ACE services that are to be notified when such commands are executed.

For instance, let's say that an ACE PTZ camera daemon is running within an ACE conference room. Also, consider a basic requirement that whenever a new person identifies him/herself at the door, that the camera point towards the door in order to visualize the new user walking into the room.

In order for this to occur, a user identification daemon (that controls the identification of users at specific ID devices) must notify the camera daemon that a new user has been positively identified at the room door and is walking in. This prompts the camera to turn towards the door and visualize the user.

The diagram below depicts the finer details of this concept:



Above a client connects to a "Service Daemon" and issues an ACE command (1). This daemon then looks up its list of commands to see if the command issued is a command that prompts a notification and if so, which existing services need to be notified for this command (2). Once the services that require a notification are looked up by the daemon, the appropriate methods in those other services are (3).

What exactly does it mean to receive a notification and have appropriate methods invoked? Furthermore, how do services get added to the notification list to begin with? When a service deems it necessary to be notified of a specific command (call this the notified service) within some other service either at startup or later (call this the notifying service), they issue an "addNotification" command to the notifying service. This effectively adds the command interface object (see section 2.4.1 above for details) of the notified service (with a specific interface method name) to the list of services to notify. Once a command is executed within the notifying service, all appropriate command interfaces of services to be notified are referenced and the listed interface methods are invoked on those services. This is what it means to receive a notification.

This capability is very useful for ACE services in that these can now react to changes in the environment and appropriately respond to user actions and adapt to environment activity, crashes, and conflicts simply by notifying each other of such changes.

More details on the design of daemon notifications can be found in the ACE Client Notification Design document.

2.5 Basic ACE Services

With the basic concept of ACE and the main daemon framework in place, basic ACE services that interact with one another and provide a basis for other higher-level services to be built upon can be constructed.

In this section, some of the basic services shall be presented. These services work together provide ACE with the accessibility, robustness, and distributiveness that define an Ambient Computational Environment. These services work together to provide the following ACE capabilities: virtual user workspaces, user registration, identification, and authentication, service discovery, data conversion and distribution, and network logging.

One of these services has already been seen – the ASD – since this was necessary to understand the basic service discovery within the daemon infrastructure. Now, some of the other services that provide the basis for the capabilities mentioned above shall be examined.

2.5.1 HRM - Host Resource Monitor

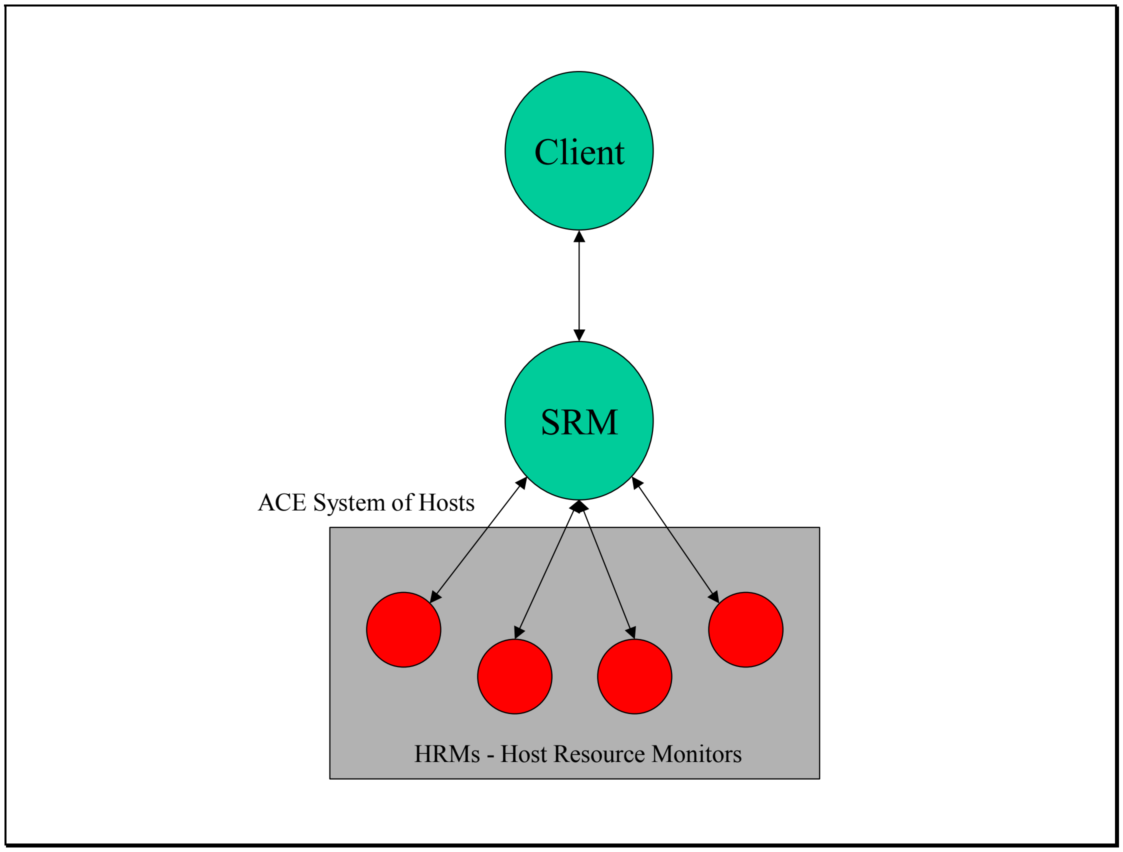
The host resource monitor service provides computational and network resource status on a single host (the same host this service runs on). The HRM provides this information in one of two possible ways. It may supply this information via notification commands (section 2.4.5) to services that request to be notified or it may be queried for specific information by client services.

Some of the information that is reported by the HRM to requesting services include host CPU load, CPU speed (bogomips), network traffic load, total and available memory, and disk storage capabilities.

See the Host Resource Monitor Design document for further details.

2.5.2 SRM – System Resource Monitor

The system resource monitor provides much the same kind of information as the HRM but is different in that it serves as the resource monitor for all the machines running in an ACE environment. As described above, it communicates with all HRMs below it in order to monitor all computing resources at a system wide level thus allowing for uniform allocation and distribution of ACE system resources. It also serves as the resource allocation interface between clients wishing to run applications within ACE and single hosts executing these applications.



2.5.3 HAL – Host Application Launcher

The Host Application Launcher is responsible for running general purpose applications on specific hosts. If a client wishes to run a certain program or application on a host, it must first connect to the HAL and request that a specific application be launched. The HAL then simply runs the requested program on its host utilizing the local host resources.

See the Host Application Launcher Design document for more details.

2.5.4 SAL – System Application Launcher

The System Application Launcher is responsible for running a specific program or application within an ACE. Much like an SRM communicates with one or more

HRMs below it to distribute resources, the SAL delegates the responsibility of launching applications within an ACE to its underlying HALs.

If an ACE client wishes to run a specific application, it requests that that be done to the SAL. The SAL then finds an appropriate HAL to launch the application (randomly or by resource allocation by communicating with the SRM) and delegates that responsibility to the chosen HAL. The HAL then runs the requested application locally on the current host.

See the ACE System Application Launcher Design document for more details on this.

2.5.5 WSS – Workspace Server

Another important ACE service is called the Workspace Server. This ACE service is responsible for creating and removing user workspaces as these get created and closed by users. It is also responsible for naming and keeping track of instances of these workspaces that are created for specific users.

More details on how this service works in order to provide users with virtual workspaces shall be given in section 2.8 where most of these services shall be integrated into some typical ACE usage scenarios.

For further details, view the ACE Workspace Server Design document.

2.5.6 ACE ID Monitor

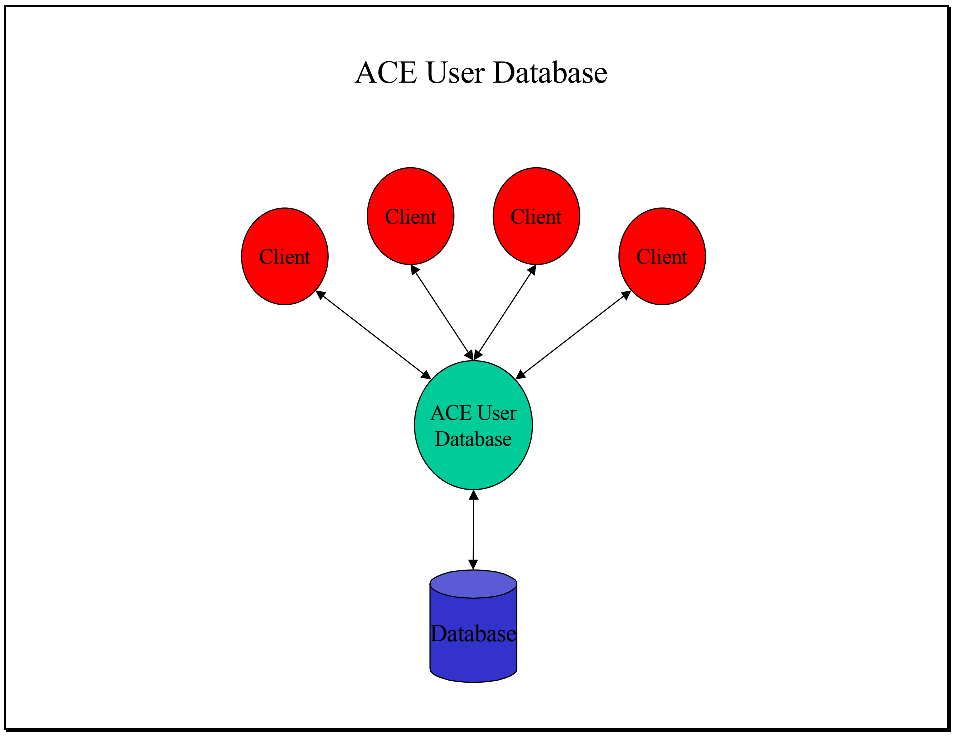
This service has the unique job of receiving user identification notifications (refer to section 2.4.5 for daemon notifications) from ACE identification devices (such as a fingerprint scanner unit service) and initiating the appropriate actions to account for a positive or negative identification notification.

For instance, let’s say a user wants to have his/her workspace show up on the screen of an access station after identifying him/herself with a fingerprint. The user presses his/her fingerprint to the scanner is positively identified. The identification device service sends a positive user identification notification to the ID monitor. This in turn causes the ID monitor to call the workspace server to start up a workspace viewer on the access station’s screen. If the user is not identified as a valid user, the ID monitor can then call the appropriate services to deal with that situation (e.g. if one wants to get extreme, call the FBI alerting service to call the FBI and alert them of an intruder that must be arrested).

2.5.7 AUD - ACE User Database

In order for any person to interact with an ACE he or she must be a valid ACE user. What does this mean? It means that the person is not only a registered user within the local computer network but also a user registered within ACE. In order to register a user with ACE, the user must register basic information such as username, password, full name, identification number (e.g. iButton #, fingerprint scan, etc), and public key (hence the reason why the user must be a valid user within the LAN) with an ACE user management service. This service should manage users and their information so that other services may or may not grant access to people wishing to utilize ACE. Such a service must communicate with yet another service called the ACE User Database. For more information on the AUD and its design components refer to the ACE User Database design document.

This service is simply an ACE interface to the database of valid ACE users and their pertinent information. The AUD takes insertion and selection query requests from other authorized ACE services to lookup and modify the database of ACE users. Access to this service and the user information is not limited to an ACE user manager but to all other authorized ACE services that wish to find out some information about a user or users they interact with.



For more details on the information maintained in the AUD refer to the AUD Database Design document.

2.5.8 ACE FIU – Fingerprint Identification Unit

This service is a simple controller interface for the Sony fingerprint identification unit model FIU-. This service communicates directly to the FIU, loading its tables of known fingerprints, querying it for identification of user fingerprints, and serving as an interface to other ACE services wishing to identify someone and/or receive identification notifications.

So, in other words, the FIU service has been implemented to identify ACE users via their fingerprints.

2.5.9 ACE IButton Reader

The IButton service is yet another user identification service that simply interfaces to the IButton reader. The IButton is a simple solid state memory device that stores a unique serial number. When associated with a user, it may be used as an identification device. This ACE service serves to read these numbers from the IButton reader, identify users based on known users and their serial numbers stored in the AUD, and interface to other ACE services wishing to identify someone and/or receive identification notifications.

See the ACE IButton User Identification Design document for further details.

2.5.10 ACE Authorization Database

This service, much like the AUD, is a database interface service that stores user and client service authorization assertions. This service is utilized by ACE services to lookup certificate assertions for users and other services attempting to execute specific commands within ACE. These assertions are passed onto KeyNote, which is used to determine if a proper assertion or chain of assertions are present thus giving the client permission to perform specific actions.

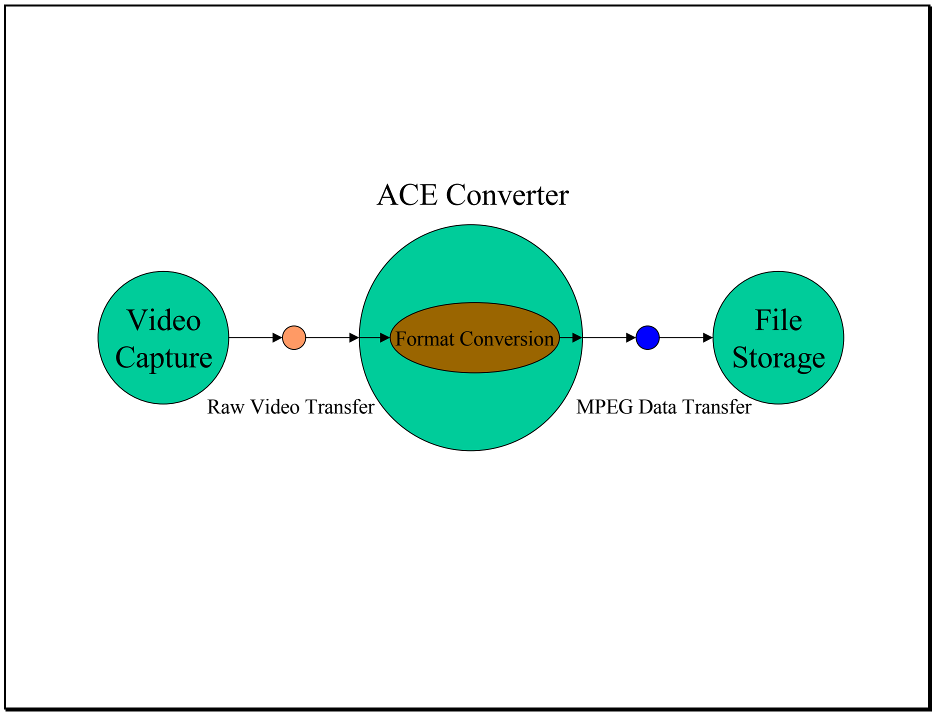
2.5.11 Room Database

In order for services to be spatially aware of their surroundings and also have some knowledge of where they reside within an ACE, their location information (along with other data) is kept within an ACE Room Database service. This service is responsible for storing and supplying services with information such as building names, room names, names of services located within specific rooms, physical location of these services in the room, physical dimensions of the room (for modeling and logistical purposes), etc… For instance, for a user to control a camera daemon and tell it to move to a specific position within the room, the camera needs to be spatially aware of its environment. That is, it needs to know where it is located with respect to locations and other objects within the room so that it may establish a 3D coordinate system for referencing the room space. Furthermore, services need to know what other services are in the room so that they may provide users with capabilities at specific locations within an ACE.

2.5.12 ACE Converter

The ACE Converter service is basically what it states it is. It performs data conversion from one format to another. For example, if video information was being transferred from a camera in the ACE to a file managing system (that is, some video information was being recorded onto disk) some kind of format conversion (or compression) should be applied before this data can be stored. In order to do so, an ACE converter is placed in between the video capture service and the file storage service. It would take the raw video stream from the camera, convert it to a format such as MPEG and send it to the file manager service for storage.

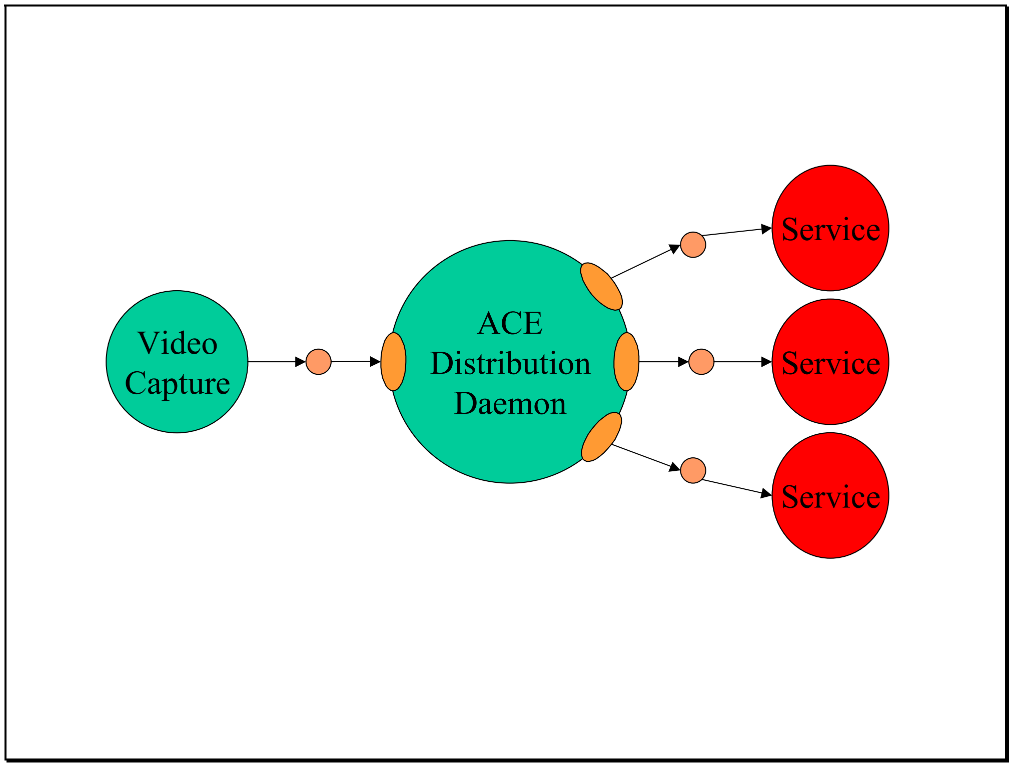
This service isn't limited to this type of conversion. It is capable of converting from one format to another from a set of known formats. A simple example is shown:



2.5.13 ACE Distribution Service

Yet another low-level data transfer service available in ACE is the ACE Distribution Daemon. This daemon is basically a data distribution service. That is, it takes in an input data stream and a set of known destination services and forwards this data onto a set of one or more other services that need or want that data.

Below is an example of a distribution daemon working to supply a set of ACE services with a video stream from a video capture service.



For more details on the ACE Distribution Daemon design, see the ACE Distribution Daemon design document.

2.5.14 ACE Network Logger

Last but not least is the ACE Network Logger. This service is used mainly for security and debugging purposes within the ACE system. This service simply stores service activity information within a set of logging files. This is used mainly to record what kinds of activities are present within an ACE system and to serve as a history so that, if necessary, system administrators can investigate them for security holes or system bugs. An example of this would be an attempt of an invalid user to log into the system. If an unknown user attempts to log into the system an invalid identification error message along with pertinent details would be sent to the ACE Network Logger for recording. If this persisted (more than one attempt in different days) proper action could be taken to prevent the user for tampering with the system.

2.6 ACE User Applications

With the basic ACE services in place, other, higher level services can then be constructed to provide users and administrators with easier access to the environment. These applications include user-friendly graphical interfaces to devices and services, legacy applications integrated with ACE, general purpose, communications, and data transfer applications, and administrative applications used to register users, delegate authority, and control ACE specific data.

In section 2.2.2, an example of a graphical user interface for controlling devices and services within ACE was demonstrated. Other legacy applications are also used in ACE. These applications were integrated and/or modified to be part of ACE. Some of these applications include VNC (Virtual Network Computing) from AT&T labs and Gnome O-Phone. Finally, some other ACE built applications have been included in the ACE system to utilize ACE resources and capabilities. These include audio & video capture and camera and projector control services.

Now, these applications shall be further developed. But before delving into these applications, how these applications can be run within ACE (temporary, restart, and robust execution) must first be examined.

2.6.1 Temporary ACE Applications

Within ACE any general-purpose application that is run on a user workspace or on an ACE machine by an ACE service is considered a temporary application. This is mainly due to the fact that such applications are not vital applications for the existence and operation of an ACE.

Examples of such applications are word processors, internet browsers, office utilities, etc. These applications are allowed to crash and it is irrelevant to the ACE system as a whole whether or not these applications are executed again.

2.6.2 Restart ACE Applications

Restart applications are those that need to be running within ACE in order for proper execution of ACE services but are allowed to crash or stop running for a small interval of time and must then be restarted.

Examples of such applications can be a user's default workspace, an ACE camera control service, the network logger (even though some activity information is lost), etc.

For this reason, these applications must be closely watched by other ACE services in order to make sure they are up and running and be restarted in case of a crash. Such a service has not yet been implemented but the ACE infrastructure makes this possible and easy to do. Notifications can be utilized to alert such watcher services of closed applications and can also work in conjunction with the ASD and the WSS to make sure such restart applications are up and running.

This type of service is the next step in our current development of ACE to ensure that applications that need to be up are always up and that they are restarted in case of a crash.

2.6.3 Robust ACE Applications

Robust ACE applications are those that are extremely vital to the proper execution of ACE services and functioning of the system as a whole. Such applications must not be allowed to crash, can be moved from one host to another with minimal to no interruption of service, or have a backup redundant instance of the application ready to take over in case it does stop running.

Such applications are ACE user management applications (that register and identify users and distribute authorization), services such as the ASD and AUD, the workspace server, etc. If any of these applications die for a long period of time the consequences could be a partial to complete halt of ACE itself.

A basis for maintaining robustness of applications within ACE, as for restart applications, has also not been fully implemented in our current version of ACE. This endeavor is a complex one and must be developed closely with the work being done within the persistent store area of ACE. This is also in its initial stage and further developments are still to come.

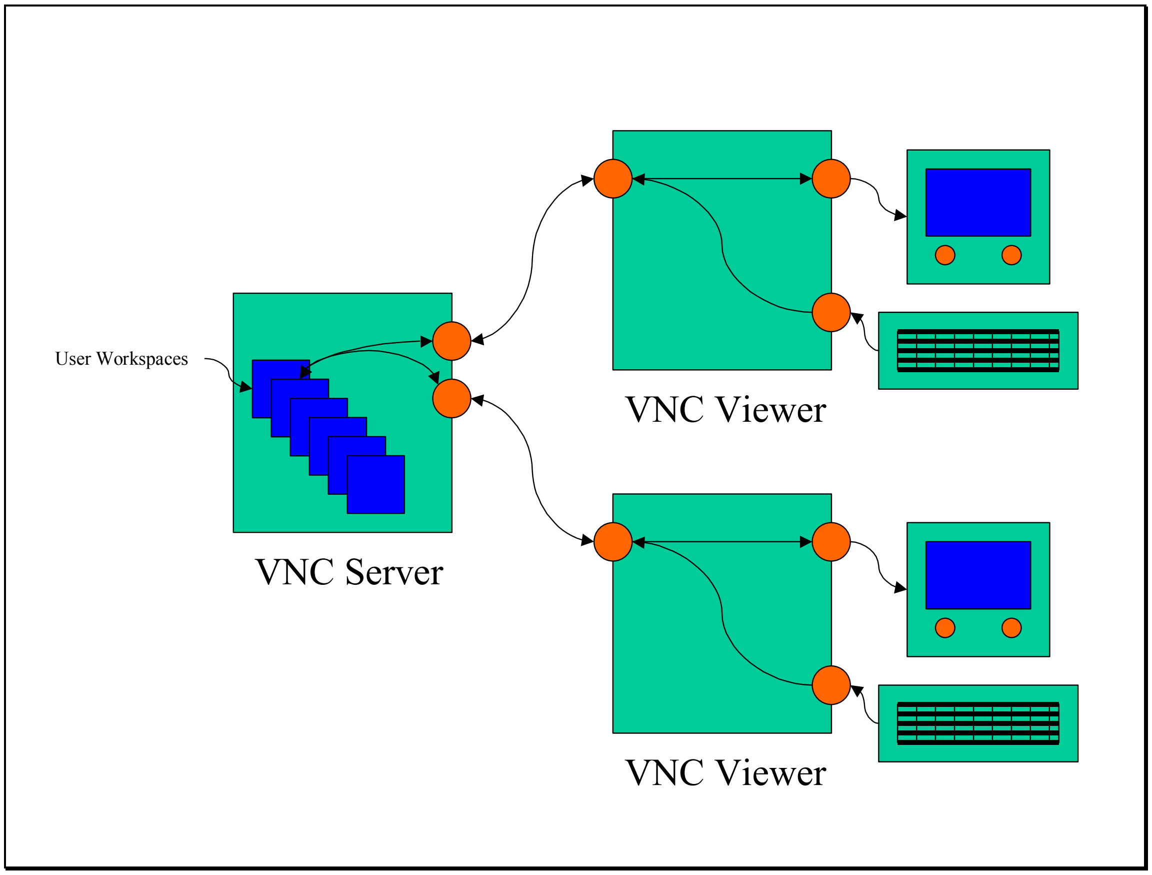
2.6.4 VNC & ACE Workspaces

For a good part of this document, the concept of user workspaces and how these are managed and utilized were discussed but no thought was given to what actually generates and controls a user's workspace. This is where VNC (Virtual Network Computing) from AT&T labs comes in.

This legacy application was taken, and its use slightly modified to fit the ACE infrastructure. VNC is used to emulate user workspaces and redirect them to appropriate ACE access points around the network.

VNC works in much like a client-server fashion. The VNC server as it is called, is responsible for actually housing or running the user's workspace, maintaining all state information and accepting input and output to the workspace when it is being viewed/utilized by a user. The VNC client, or VNC viewer as it is called, is simply a client program that runs remotely on a simple network access point and connects to the VNC server. The server then redirects all I/O to that client/viewer thus allowing the user to remotely see and command his/her workspace that is running at the server location. Of course that in order for someone to access a workspace (i.e. run the VNC viewer) he/she must provide the proper user password to the VNC server so that proper access can be given. VNC also allows the user to have more than one running workspace at the VNC server. This way a user may have multiple workspaces for running different applications.

The diagram below shows a simple schematic of how VNC works by redirecting I/O to remote machine viewers.



VNC usage was slightly modified for ACE use. As seen before, the WSS is responsible for managing user workspaces. It creates them, names them based on whose workspace it is and where it is running, and deletes them when needed. It must also verify user VNC passwords so that only valid users may see and access their running workspaces. The normal execution of a VNC viewer requires that the user input his/her password directly to VNC. Unfortunately, within ACE, the VNC sessions are managed and controlled by the WSS. For this purpose, the VNC password files were directly access and modified by the WSS when new workspaces were created and when users accessed their workspaces from remote access points. This guaranteed that the password verification by VNC was made invisible to the normal ACE user. Simply by being a valid ACE user and properly identifying him/herself with one of the ACE identification devices, the user is allowed to access his/her workspaces via the WSS.

As was mentioned before, more insight shall be given to ACE workspaces and how these are run in section 2.8 below.

2.6.5 O-Phone & ACE

Another open source, legacy application incorporated into ACE is called the O-Phone. This application enables full-duplex telephone communication over IP. Thus allowing users to call each other and even outside ACE phones from their workspaces.

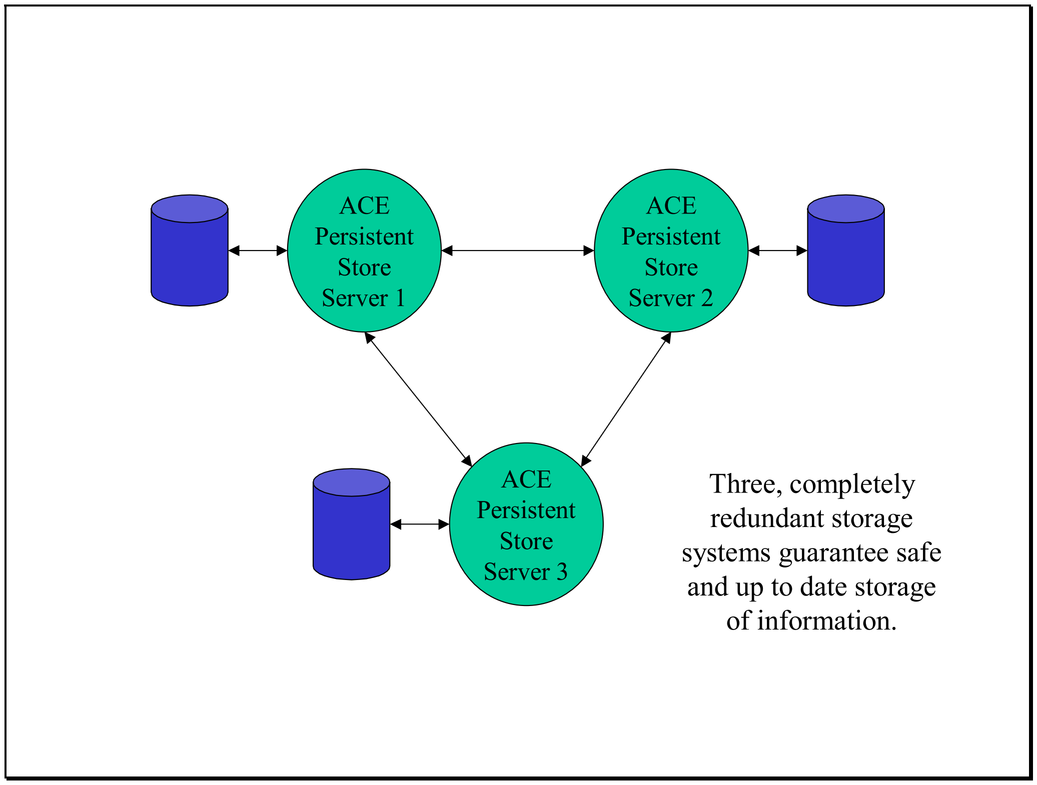
This frees up the user from having to be near a phone to make a call. If a valid ACE user is near an access point, he/she can bring up a workspace and make a phone call.

No major modifications were done to this application in order to incorporate it in ACE. It was merely adapted to work properly with our local network and can thus be run from any user workspace as a regular system application. No GUI has yet been developed for this application for ease with ACE usage.

2.7 ACE Persistent Store

For ACE to be deemed the robust and long-lived user environment for both data and running applications, a method of securely storing and retrieving data and application state information and guaranteeing their persistence needs to exist. For this reason it became necessary to develop persistent store architecture for ACE. See the ACE Persistent Store Requirements document for more information on specifics persistent store needs.

The initial framework entails the use of three completely redundant and interconnected server systems.


ACE Persistent Store Server 1
ACE Persistent Store Server 2
ACE Persistent Store Server 3
Three, completely redundant storage systems guarantee safe and up to date storage of information.

The three storage systems above work together to constantly update each other's data at specific intervals. In doing so, this system ensures that the same exact data is stored within each of their individual storage areas.

If for any reason, one or two of the servers crash or are taken down, ACE services may still access the stored information within them. Furthermore, by having three separate storage servers, it is possible to remove potential bottlenecks of many ACE services attempting to access information on a single storage server at the same time.

ACE utilizes this redundant storage systems in order to safely maintain and always have available all the user, service, and configuration information necessary to make ACE a robust architecture on which services can run.

For more details, see the ACE Persistent Store Design document.

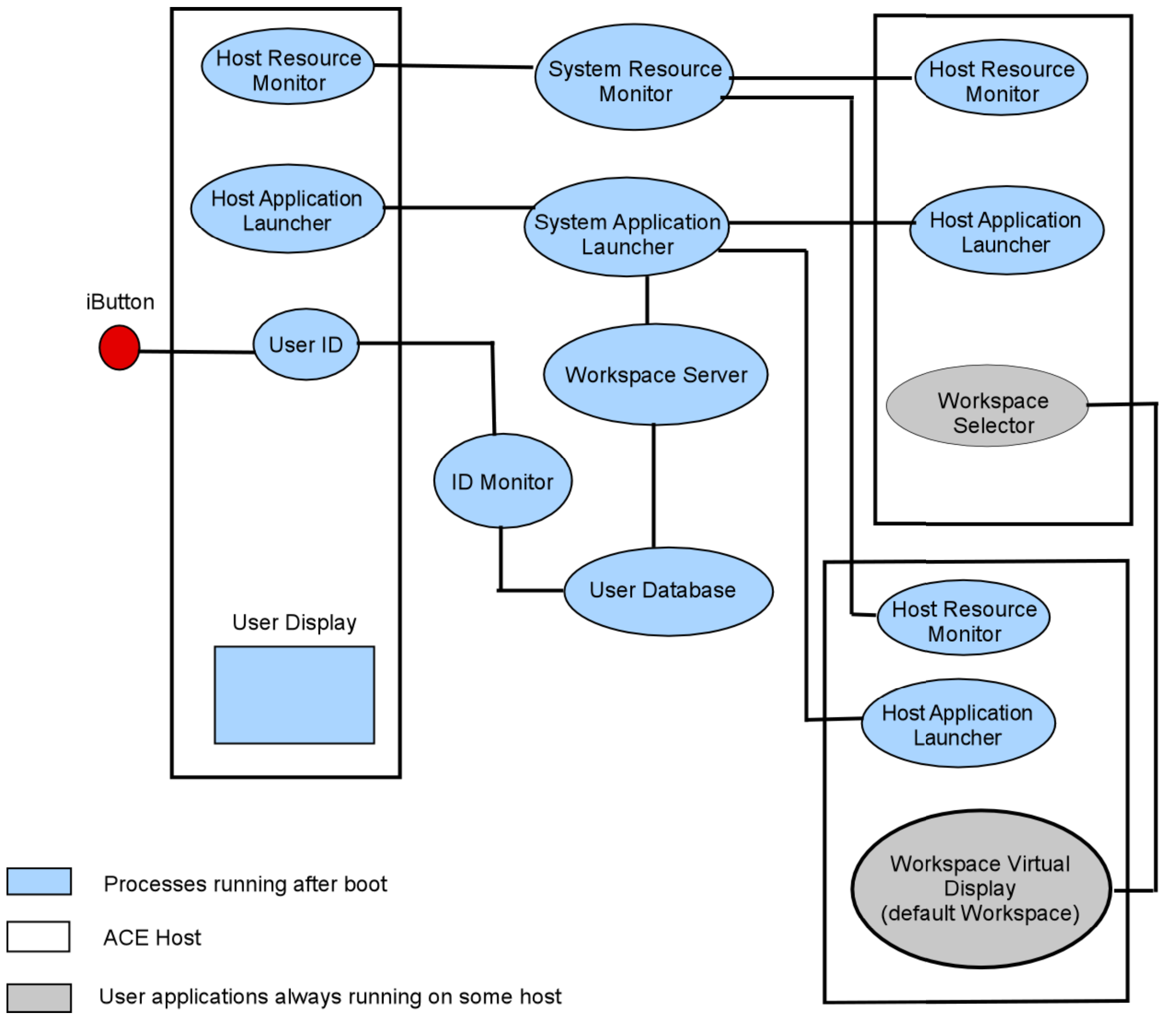
2.8 ACE Scenarios

After touching on the ACE architecture and the main design components that work together to make ACE a robust and interconnected framework of services, some ACE scenarios can now be developed. These scenarios shall illustrate how the pieces described above fit together to supply a coherent set of services and capabilities.

2.8.1 Scenario 1 – New User & User Workspace

John Doe is a new employee at ACECo. After getting settled in and getting to know his new work environment he presents himself to the system administrator to have a new ACE user account established for himself.

The administrator obtains from him all the pertinent information and creates a new Unix user account for John. Then, he adds this new user as a new ACE user as well. Utilizing a simple GUI, the administrator inserts John and his new account information into the user database via the AUD service. His fingerprint is scanned and added as well. As John is being added as a new user, the GUI also communicates with the Workspace Server. The WSS verifies that this is a new user by checking that the user location information within the AUD is empty. This prompts the WSS to create a new VNC server session (i.e. the user's default workspace) for John. To do so, the WSS requests from the SAL that a new VNC session for user John be started somewhere. The System Application Launcher works in conjunction with the HAL, SRM, and HRM to create a new default workspace for John. First, the SAL inquires with the SRM to find out which machine in the ACE network is most suitable (has the most free resources or is best for a given application) for running the VNC server application. The SAL has this information with it from the regular communications it holds with all the HRMs in the network to obtain host resource information. Once a suitable host has been selected, the SAL requests from the HAL on the selected host to launch a VNC server application as the default workspace for user John. The diagram below shows the connections between the services for a better understanding of these steps.


Host Resource Monitor
System Resource Monitor
Host Resource Monitor
Host Application Launcher
System Application Launcher
Host Application Launcher
iButton
User ID
Workspace Server
Workspace Selector
ID Monitor
User Database
User Display
Host Resource Monitor
Host Application Launcher
Workspace Virtual Display (default Workspace)
Processes running after boot
ACE Host
User applications always running on some host

With this being done, John now has a default workspace constantly running on the selected host.

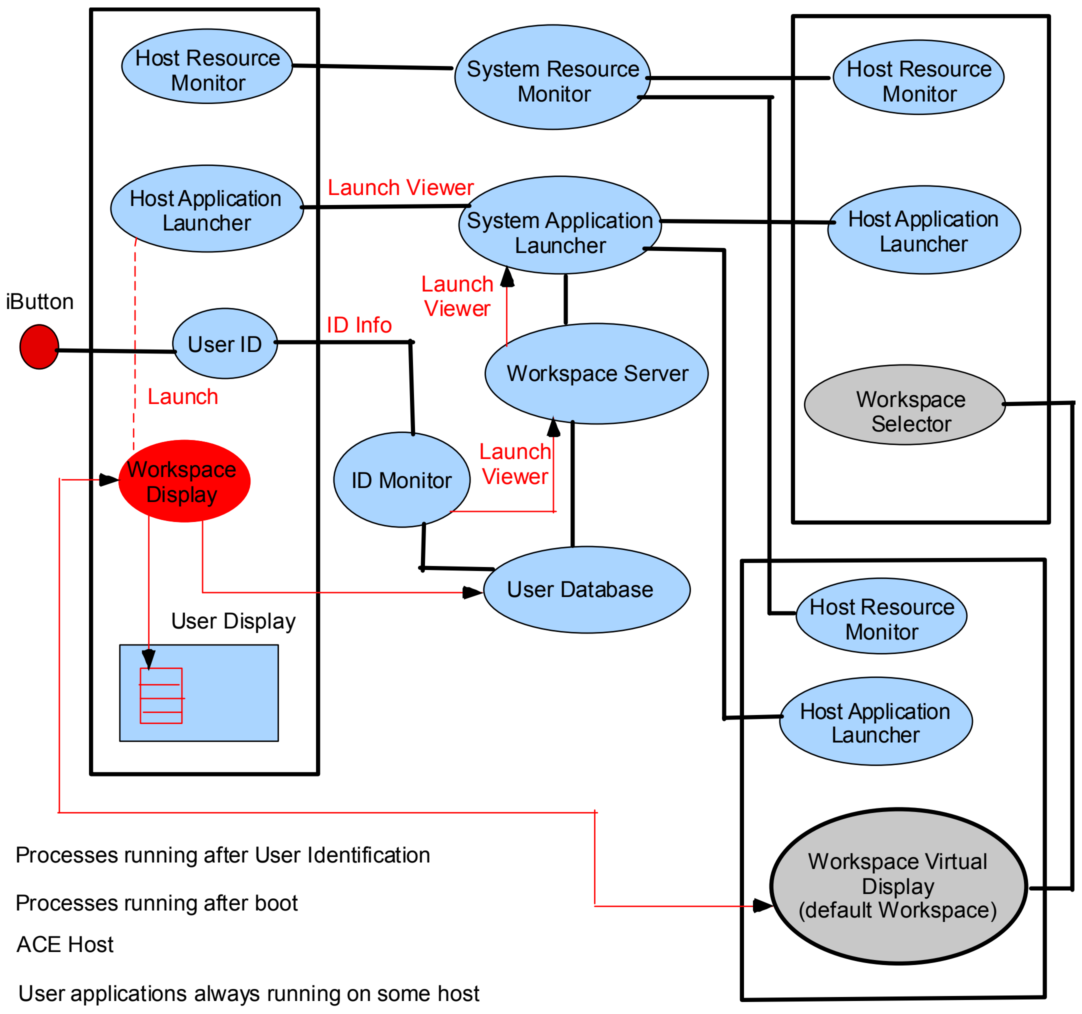
2.8.2 Scenario 2 – User Identification

John is to give his presentation in the ACE conference room in 30 minutes. In order to prepare for his presentation he goes to the conference room and identifies himself at the podium computer by pressing his thumb to the fingerprint scanner next to it. As soon as this is done, the user ID daemon, which constantly polls the FIU (Fingerprint Scanning Unit) daemon, requests from the FIU that the user be verified. In John's case, he is a valid ACE user and is thus positively identified. This causes the User ID service to send out a notification that user 'John Doe' has been identified at the ACE conference room to the ID Monitor service. The ID Monitor service then updates John's current location with the AUD. At this point John Doe has been successfully identified and other services can then be invoked.

2.8.3 Scenario 3 – User Workspace

Once John Does has been successfully identified, the ID Monitor service is responsible for communicating with the Workspace Server so that the user's workspace can be brought up to the user's current access point.

The WSS invokes the SAL to start a VNC viewer session at John's current location (which was obtained from the ID Monitor) so that his default workspace may be brought up. The SAL then invokes the HAL on John's current host. This in turn starts John's VNC viewer application so that he may access his default user workspace. Effectively, John's workspace pops up on his screen at the conference room podium and he goes on to access his presentation files and setup his speech. The diagram below shows the entire process from user identification to displaying the user workspace. The red circle on the left labeled iButton is where the user would identify him/herself (the fingerprint scanner in John's case).



🟥 Processes running after User Identification

▨ Processes running after boot

□ ACE Host

▨ User applications always running on some host

2.8.4 Scenario 4 – Multiple User Workspaces

Can a user have more than one workspace? The answer is yes. ACE users are not limited to a single workspace (as has been mentioned earlier in this document). The default workspace is created for all users so that he/she may have at least one valid and working workspace through which to access the network and its resources.

In creating new workspaces, the same basic ACE services are utilized to launch, maintain, and display the running workspaces. The only significant difference is that now users are presented with a workspace selector when a user identifies him/herself. This selector works in conjunction with the WSS to find and display a workspace selected by the user. The diagram above also shows how the workspace selector fits within the process.

Let's say that John, has been working on his presentation from a separate workspace that is not his default workspace. So now, John has 2 valid running workspaces. As soon as he identifies himself a small workspace selector GUI pops up on the access point screen showing him a list of all workspaces running and available. He selects the secondary workspace he owns and that causes the WSS to attempt to launch the viewer of this workspace. It communicates with the SAL to do so.

Soon enough, John's secondary workspace is up and visible on the podium access point and John continues to setup his presentation.

2.8.5 Scenario 5 – ACE Services & Devices

One last thing needs to be done before John can get his presentation ready to go. He must turn on the projector and set the camera to point towards the podium. Using his workspace, he starts the ACE device GUI. This GUI communicates with the ACE Room Database service to find out what devices are present within the room, where they are, and what current settings they have.

John selects from a graphical representation of these devices, the projector. This pops up another screen that shows the different controls that can be set for the projector. He uses it to turn the projector on and to output the workspace to the screen. This effectively causes the GUI program to communicate with the projector daemon and send appropriate messages to execute John's commands. Then, he selects the camera output to stream to the projector as a picture in picture output. Finally, he selects the camera and uses it to control the camera and pan, tilt, and zoom it towards the podium.

All of these commands performed by John are fundamentally simple daemon to daemon communications that are used to execute actions. Client and daemons find each other via the ASD and communicate with each other to obtain and supply the proper information to perform a given task.

John is now ready to give his presentation.

3.0 Design Constraints

<Any known constraints on design elements here>

4.0 Deprecated Designs

5.0 Glossary

<Refer to a project glossary for terms used throughout the project>

6.0 Change Log

Version	Date	Person	Reason
0.1	July 12th, 2001	Renzo Hayashi	Original
1.0	December 13th, 2001	Renzo Hayashi	First complete version

7.0 Notes