Universal Agent 7.7.x Administration

Universal Agent 7.7.x

Table of Contents

1 Overview

Universal Agent 7.7.x Administration provides information on Universal Agent Databases and Universal Agent Security for all Universal Agent packages on all supported operating systems.

The following information is provided for the administration of Universal Agent components:

Universal Agent Databases	Universal Agent components provide features such as fault tolerance, managed configuration, event subsystem (UES) data collection, and event monitoring that rely upon a set of databases for their implementation: • Component Information Database • Universal Event Monitor Databases • Universal Enterprise Controller Databases • OMS Server Message Database • Database Backup and Recovery • Database Recovery for Universal Broker • Database Recovery for Universal Enterprise Controller • Listing Universal Agent Database Records Examples • Removing Universal Agent Database Records Examples
Universal Agent Security	Each Universal Agent component is designed to be a secure system. Security is comprised of the following basic elements: • Security of Universal Agent Components • Universal Access Control List (UACL) • UACL Examples • X.509 Certificates • Creating Certificates Examples

2 Licensing

Prior to Universal Agent 6.9.0.0, licenses for individual Agent components were stored in their respective configuration files. Aside from verifying that the license information was valid (that is, the individual license parameters were capable of generating a key at runtime that matched the configured license key), no enforcement of license counts was done.

With Universal Agent 6.9.0.0, the method by which license information is obtained and configured changed.

To enable centralized visibility and enforcement of Universal Agent licenses, Universal Controller 7.7.0.0 provides all license information to distributed Agents via a Universal Controller for Agent Management (UCAM) deployment,. This provides an opportunity to configure and enforce licensed Agent counts and manage the use of optional Universal Agent components.

3 Universal Agent Databases

3.1 Introduction

Some Universal Agent components provide features that rely upon a set of databases for their implementation. Such features include fault tolerance, managed configuration, event subsystem (UES) data collection, and event monitoring.

Unless otherwise noted, the Universal Broker owns all databases and performs all direct database access. The Universal Broker processes and responds to all database access requests it receives from individual Universal Agent components.

3.2 Detailed Information

The following pages provide detailed information for Universal Agent Databases:

- Component Information Database
- Universal Event Monitor Databases
- Universal Enterprise Controller Databases
- OMS Server Message Database
- Database Backup and Recovery
- Database Recovery for Universal Broker
- Database Recovery for Universal Enterprise Controller
- Listing Universal Agent Database Records Examples
- Removing Universal Agent Database Records Examples

3.3 Component Information Database

3.3.1 Component Information Database

The component information database records information about all Universal Agent server components that the Universal Broker manages. It is opened during Universal Broker start-up processing.

The information captured by the Universal Broker includes, but is not limited to, the component's process ID, start time, current state, and end time.

One important aspect of this database is its ability to record the current state of a Universal Agent server component. Each time a component's state changes, it sends a notification to Universal Broker, which updates that component's record.

For the Universal Agent components that offer it, this component state provides the basis for reconnect or restart functionality, otherwise known as network or manager fault tolerance (for Universal Agent: see Fault Tolerance Implementation or network fault tolerance (for Universal Data Mover: see Network Fault Tolerance - Universal Data Mover.

When a Universal Agent server process finishes executing and its component state indicates that it has completed, Universal Broker deletes that component's information from the database.

Universal Broker stores Universal Agent component information in the **bcomponent.db** and **scomponent.db** database files.

IBM i	The database, **UBR_CMP_DB**, is located in the spool library, **UNVSPL620**.
UNIX	The database file resides in the **/var/opt/universal/spool** directory.
Windows	The database file default location is `C:\Program Files\Universal` (64-bit Windows systems)
z/OS	Universal Agent components access this file via an HFS- or zFS-allocated dataset, which is mounted on the z/OS Unix System Services (USS) file system. Universal Broker is capable of dynamically mounting this database during start-up, if it is not already mounted.

3.3.2 Universal Spool

The Universal Spool - or more simply, the spool - supports storage of the standard file I/O that Universal Command Server redirects from user processes. When Universal Command Server executes with spooling enabled, it captures a user process' standard input, standard output, and standard error and writes them to the spool.

Universal Spool is implemented as a set of databases. Universal Broker and Universal Command Server remove the database records automatically when they are no longer required. No database maintenance jobs are required.

Universal Broker and server components are the only programs that access the spool. No user access is required. The operating system's file system security should be used to prevent all access to the spool except for the broker and server. The Broker and server require full control permissions to the spool in order to add, delete, update, and read database files.

All standard I/O files written to the spool are encrypted to insure data privacy.

The spool database files that store redirected standard input, standard output, and standard error have the format **spool.stdin.compid.db**, **spool.stdout.compid.db**, and **spool.stderr.compid.db**, respectively. In each of the names, **compid** is replaced by the server's actual component ID.

IBM i	These files are located in the spool library, **UNVSPL520**, in the following format: • **SExxxxxxxx** (**SE** = standard error) • **SOxxxxxxxx** (**SO** = standard output) • **SIxxxxxxxx** (**SI** = standard input) In each file name, **xxxxxxxx** is the component ID expressed in hexadecimal. (The component ID is the Component ID used to identify a Stonebranch process. This is the same Component ID output by Universal Query.)
UNIX	These files reside in the **/var/opt/universal/spool** directory.
Windows	The database files' default location is: `C:\Program Files\Universal` (64-bit Windows systems)

z/OS	Universal Agent components access these files via an HFS- or zFS-allocated dataset, which is mounted on the z/OS Unix System Services (USS) file system. Universal Broker is capable of dynamically mounting this database during start-up, if it is not already mounted.

3.4 Universal Event Monitor Databases

3.4.1 Universal Event Monitor Databases

To support the Universal Event Monitor (UEM) component, the Universal Broker provides and manages the following databases:

- Event Definition Database
- Event Handler Database
- Event Spool Database

These database files are local to each system. The Universal Agent install script is responsible for creating the database directory. If the Universal Broker attempts to open a database file that does not exist, it will create that database.

UNIX	The default database directory is `/var/opt/universal/spool`.
Windows	The default UEM database directory is: • `C:\Program Files\Universal` (64-bit Windows systems)

Note

UEM Server is only available for Windows and UNIX. The UEM databases are used only on those operating systems.

3.4.2 Additional Information

The following page provide additional information for Universal Event Monitor Databases:

- Controlling Universal Event Monitor Database Access

3.4.3 Event Definition Database

3.4.3.1 Event Definition Database

The Universal Event Monitor (UEM) Server stores information about the events that it monitors in the event definition database.

An event definition record describes a system event and provides the information that UEM uses to track an event occurrence and test for its completion. An event definition record also may contain information that UEM uses to respond to (that is, "handle") an event's successful completion, its failure to complete, and even its failure to occur.

An event-driven UEM Server relies upon stored event definitions for its input. When an event-driven UEM Server starts, it asks the Broker for all event definitions assigned to it. If no event definitions are assigned to a particular event-driven Server, that Server continues to execute but will not do any actual event monitoring.

A demand-driven UEM Server also may obtain its input from a stored event definition record, but it is not required. Typically, a demand-driven Server receives its input from the UEM Manager's command line parameters.

Event definition records are added and maintained with the UEMLoad Utility.

3.4.4 Event Handler Database

3.4.4.1 Event Handler Database

An event handler record describes the action that Universal Event Monitor (UEM) should take in response to a monitored event's outcome. This action, or response, is simply a system command or script that UEM executes upon an event's completion, failure to complete, or failure to occur. The UEM Server executes these processes in a secure context, using user account credentials stored in the event handler record.

A Stonebranch Tip

Security is a primary concern within all Universal Agent components.

Whenever the user account information stored in an event handler record includes a password, that password is encrypted using the Data Encryption Standard (DES) algorithm.

An event-driven UEM Server relies upon stored event handlers to determine its response to the events that it monitors. The event definition records that describe events to UEM also contain the IDs of event handler records that UEM should use to respond to those events.

A demand-driven UEM Server also may respond to an event using a stored event handler record, but it is not required. Typically, a demand-driven Server relies upon the UEM Manager's command line parameters to describe the actions that it should take in response to the event that it monitors.

When a UEM Server needs to use a stored event handler record, it sends a request to the Universal Broker to retrieve the record using the ID specified in the event definition record or provided from the UEM Manager command line. The Universal Broker returns the event handler record to the UEM Server, which then executes the specified system command or script.

Event handler records are added and maintained with the UEMLoad Utility.

3.4.5 Event Spool Database

3.4.5.1 Event Spool Database

Universal Event Monitor (UEM) records its monitoring activity in the event spool database.

It is possible for UEM to detect multiple occurrences of any single event that it monitors. UEM creates a record in the spool database for each event occurrence that it detects and tracks. UEM maintains the current state of an event occurrence from initial detection through the completion of any event handlers.

If an event definition goes inactive before UEM detects any occurrences of that event, UEM creates a single spool entry to record the expired event.

Universal Broker applies all updates to the event spool database. A UEM Server is responsible for sending the Universal Broker all relevant information, along with the required database operation (add, update, or delete).

Typically, any spool records created for an event are deleted when the Broker detects the completion of the UEM Server that monitored the event. However, when an event-driven UEM Server completes, any records that indicate work in progress (for example, tracking of an event occurrence, execution of an event handler) are retained for possible recovery when the event-driven Server is restarted. For additional information on recovery of event spool records, see Universal Event Monitor Server.

A Stonebranch Tip

An option can be set in the Universal Broker's configuration to prevent it from deleting any event spool records when the UEM Server component completes. Setting the **comp_info_retention** option to a value greater than 0 causes the event spool record to be preserved.

Because there is currently no database cleanup routine available, this option should be set only following a recommendation from, and with the assistance of, Stonebranch Inc. Customer Support.

Feedback from a demand-driven UEM Server is returned to the UEM Manager that initiated the monitoring request. In this situation, event spool records are simply another means of following the progress of the event and any detected occurrences.

However, for an event-driven UEM Server that has no client, the records in the event spool database are the best way to monitor the status of the work performed by that UEM Server. Because an event-driven UEM Server typically is a long-running process, an adequate history of the UEM Server's activity can be obtained by viewing the spool records.

Currently, event spool records can only be viewed with the Universal Spool List utility (**uslist**). Information on using Universal Spool List to view event spool records can be found in Universal Event Monitor Server. For information on all Universal Agent utilities, see the Universal Agent Utilities 7.7.x Reference Guide.

3.4.6 Controlling Universal Event Monitor Database Access

3.4.6.1 Controlling Universal Event Monitor Database Access

Universal Broker is responsible primarily for providing access to the Universal Agent databases. However, there are utilities provided, including the Universal Spool List (**uslist**) and Universal Spool Remove (**uslrm**), that can be used to access the databases directly. While these utilities should be used only following a recommendation from and with the assistance of Stonebranch, Inc. Customer Support, they are documented fully in the Universal Agent Utilities 7.7.x Reference Guide.

To protect the database contents, operating system permissions on the database files themselves should be set so that only accounts with super-user or administrative privileges has access to them.

UEM provides its own command line utility, UEMLoad, to maintain the event definition and event handler databases. While the contents of these databases can be viewed using the Universal Spool List utility, it is recommended that all access be done using UEMLoad. The ability to remove event definition and event handler records is only provided with UEMLoad.

UEMLoad only can manage event definition and event handler databases that are local to the system on which it resides. To process a request, UEMLoad sends a request to the Universal Broker running on that system to start a demand-driven UEM Server. Next, UEMLoad sends the database request to the UEM Server, so that the UEM Server can validate the request and provide any required default values. The UEM Server then forwards the request to the Universal Broker, so that the changes can be applied to the appropriate database.

UEMLoad executes in the security context of the user account that started it. Since the Universal Broker applies changes to the event definition and event handler databases, any user with the authority to execute UEMLoad will, effectively, have access to a

secure resource. It is therefore strongly recommended that the privileges on UEMLoad be set such that only those user accounts with super-user or administrative privileges be allowed to execute it.

Application support also is provided to further limit access to the event definition and event handler databases. A type of Universal Access Control List (UACL) is provided by UEM to grant or deny local user accounts the authority to access these databases.

To fully secure the event definition and event handler databases, a UACL entry can be defined to deny access to all user accounts. Then, additional entries can be defined to grant database access to those user accounts with the appropriate authority. Whenever UEMLoad is executed, the entries in the UACL will be checked. If a match cannot be found which indicates that the user account that started UEMLoad is allowed to access the database, the application will terminate with an error.

Universal Access Control List (UACL) provides a more thorough overview of the UACL feature. For information on the specific UACL used to control access to the event definition and event handler databases, see the DATABASE_MAINTENANCE_ACL UACL entry in the Universal Event Monitor 7.7.x Reference Guide.

The event spool records generated by a UEM Server only can be viewed with the Universal Spool List utility.

3.5 Universal Enterprise Controller Databases

3.5.1 Overview

Universal Enterprise Controller (UEC) uses databases to maintain agent, user, configuration, and event data.

3.5.2 Database Files

The UEC databases reside in three files:

Database Name	File Name	Contents
UEC database	`uec.db`	Definitions of agents, groups, users, SAP systems, and a record of updates to distributed components' configurations in a managed environment.
UEC transient database	`uec.tmp.db`	UES monitor events and component information that is temporary to support I-Activity Monitor. This file is deleted and created upon restart of UEC.
UES database	`uec.evm.db`	Universal Event Subsystem (UES) persistent events.

3.5.3 Database Management

3.5.3.1 Automated Database Cleanup

Two routines are run to clean up records that meet their expiration criteria from their UEC database.

1. Routine for monitor events used for I-Activity Monitor.
2. Routine for persistent events stored for the Universal Event Subsystem.

Both routines execute at UEC start-up. Thereafter, they are scheduled to execute one hour after the previous execution's completion. At the time of execution, all records that meet the expiration criteria are removed from their UEC database.

The following UEC configuration options control database record retention:

- COMMIT_COMPLETE_EXPIRATION
- COMMIT_INCOMPLETE_EXPIRATION
- MONITOR_EVENT_EXPIRATION
- PERSISTENT_EVENT_EXPIRATION

3.5.3.2 Memory Management

Berkeley DB uses a temporary cache in memory to manage its databases. If this cache becomes sufficiently large, it must be written to disk.

Berkeley DB has a default location for storing temporary cache files, but if UEC cannot access that location, or there is no space to write these files in the default location, the following error can occur in UEC, and UEC shuts down:

```
UNV5101D Database error: 'temporary: write failed for page XXXXX'
```

To work around this issue, the following steps will write the temporary cache files to the UEC database directory:

Step 1	For z/OS installations, mount the **UECDB** HFS or zFS dataset.
Step 2	Inside the UEC database directory (or, on z/OS, the mount point), create a text file named **DB_CONFIG**.
Step 3	Inside the **DB_CONFIG** file, add the following string: `set_tmp_dir *dbpath*` In this string, `dbpath` is the path to the location in which the database files reside.
Step 4	Start / restart UEC.

3.6 OMS Server Message Database

- Overview
- Message Database Structure

3.6.1 Overview

Universal Message Service (OMS) provides reliable message delivery between OMS clients, even in the case of system failures. Message reliability is accomplished, in part, with a persistent OMS message database for saving messages to persistent storage. The OMS message database ensures that each message written to the database is flushed to disk before considering the message successfully saved.

3.6.2 Message Database Structure

The database is composed of one or more message data files and a message index. The message data files are sequential files that contain the OMS messages. The message data files keep a journal of all message operations, such as adding a message, delivering a message, and deleting a message. The message index is high-performance index into the messages contained in the message data files. The index is always created anew when the OMS Server starts.

3.6.2.1 Message Data File

Message data files are sequential files in which all message I/O operations are maintained. Message I/O operations are represented as records, which are always appended to the active message data file. Consequentially, the message data file is logically an ordered journal of message I/O operations.

The size of a message data file will grow to a maximum size that is configured with the MAX_DATA_FILE_SIZE OMS configuration option, which defaults to 25 megabytes. When the active message data file maximum size has been reached, the message data file is closed, reopened for read-only access, and a new active message data file is created.

Message data files are deleted by an OMS Server clean-up process that runs periodically or when the OMS Server starts as part of the database validation. The OMS Server message data file clean-up process will analyze all the message data files to determine if they contain any undelivered messages. If an undelivered message is found in a message data file, that message data file - as well as any subsequent message data files - will not be deleted. When all messages have been delivered, the message data file is deleted. The time interval in which message data file clean-up is performed is configured with the MSG_CLEANUP_INTERVAL OMS configuration option, which defaults to 30 minutes.

3.6.2.2 Message Index

The message index file is a keyed index into the message data files used, which provides high-performance access to the OMS messages. The message index is created each time the OMS Server starts.

3.6.3 Message Database Environment

The OMS message database location is specified by the SPOOL_DIRECTORY OMS configuration option. The default directory is platform-dependent.

Platform	Default Location
Windows	`C:\Program Files\Universal\spool\oms`
UNIX	`/var/opt/universal/spool/oms`

The message database can reside on a network file system or a SAN. The only requirement is that when the OMS Server executes on a UNIX platform, the file system must support POSIX advisory file locks when the OMS Server is part of a High Availability cluster.

The message database directory contains the following files:

File Name	Description
`omsmsg-`*`SEQ`*`.data`	Message data files, where `SEQ` is a 10 digit sequence number.
`omsmsg-`*`SEQ`*`-v`*`VER`*`.archive`	Message data file archive created when a corrupted message data file is recovered. `SEQ` is a 10-digit sequence number, and `VER` is a 4-digit archive version number.
`omsmsg.index`	Message index file.
`omsmsg.lock`	Message database lock file.

3.6.4 Message Database Performance

OMS Server throughput depends largely on the performance of the I/O subsystem on which the message database resides. Each record written to the message data file is flushed to disk, so buffering has no effect on performance. For that reason, the OMS Server message database should reside on a sufficiently fast I/O subsystem. Precise hardware configurations are beyond the scope of this document. The OMS Administration Utility does include an I/O test command that can be used to test the performance of an I/O subsystem.

3.6.5 Message Database Portability

Message data files and the message index are not portable between different platforms. The data stored in the message files contain data types that are formatted according to the platform on which they were created. For example, if the OMS Server is executing on a UNIX 32-bit, big-endian platform, the message data files cannot be moved to a UNIX 64-bit platform, nor to a UNIX little-endian platform. Generally, the message database files must not be moved between different operating systems or hardware architectures.

3.6.6 Message Database Reliability

The OMS message database can sustain system interruptions without loss of data. When the OMS Server starts, it performs message database validation on all message data files. If a corrupted message data file is found, it is truncated at the last good record. Prior to recovering a corrupted message data file, an archive version of the file is created with an extension `.archive`.

A message data file can become corrupted for a number of reasons, but the most likely reason is an uncontrolled shutdown of the OMS Server process or the host on which the OMS Server is executing. In the case of an uncontrolled shutdown during a write operation, since all I/O operations on the message data files are append operations, only the last record being written at

the time of the interruption will be corrupted. Message database validation will detect the corrupted record and truncate the file to removed the corrupted record. Since the corrupted record was never successfully written, the OMS message was never considered to have been successfully persisted. The OMS client that produced the message will resend the message when its OMS server connection has been reestablished.

The message database cannot be recovered from a corrupted file system or disk sectors. For this reason, the message database should reside on a reliable, fault-tolerant file system and hardware configuration, such as a RAID configuration, in order to maintain database reliability in the event of a hardware or operating system fault. The message database cannot be backed up for reasons described below. A reliable, fault tolerant file system and hardware configuration is highly recommended.

3.6.7 Message Database Backup

The message database should not be backed up. The reason is that a restored backup can potentially contain undelivered OMS messages that were actually delivered since the time the backup was made. The OMS Server would redeliver the message, potentially resulting in many duplicate messages being delivered to OMS clients with unpredictable consequences.

3.7 Database Backup and Recovery

3.7.1 Overview

Universal Agent databases, on operating system's other than IBM i, are implemented using Oracle's Berkeley Database product.

Recovering from database corruption requires the following steps:

1. Dump the corrupted database to a file using the Universal Database Dump utility.
2. Reload the database from the dump file using the Universal Database Load utility.

Database corruption can occur if the system or address space that is managing the databases ends abnormally. A Universal Agent program that utilizes databases should not be terminated abnormally.

Abnormal methods of termination include:

- z/OS CANCEL or FORCE command.
- UNIX SIGKILL signal.
- Windows process termination through the Task Manager.

3.7.2 Database Backups

Database recovery is not a replacement for database backups. If the data maintained by the product in the database has long term value, the databases must be periodically backed up.

3.7.3 General Database Recovery Procedures

Generally speaking, database recovery follows the same steps independent of platform and database file.

Multiple attempts may be necessary in order to successfully recover from database corruption. Stonebranch Inc. recommends that you begin with the least aggressive recovery method and only proceed to more aggressive methods if necessary.

For the first recovery attempt, execute the Universal Database Dump utility with the **-r** (lowercase) command line option. This option instructs the utility to recover as much data as possible. Depending on the extent of database corruption, this may result in a recovered database with some incomplete key/data pairs.

Reload the database using the Universal Database Load utility, and specify the **-o** option. This option instructs the utility to remove the underlying database file, which results in a clean reload from the dump file.

If the database passes validation when you restart the application, it is likely that all data was successfully recovered and no additional recovery attempts are necessary.

If the database fails validation, rerun the Universal Database Dump utility and omit the **-r** option. This results in a dump of only the most complete data. While this improves the chances for successful recovery, some data loss is likely. Rerun the Universal Database Load utility and restart the application.

If both recovery attempts fail, you may delete the corrupted database and restart the application. This results in a total loss of data, but will allow the application to execute. The application will create the missing database during startup.

3.8 Database Recovery for Universal Broker

3.8.1 Database Recovery for Universal Broker

Universal Broker uses databases to maintain component information, configuration information, and event data. A corrupted database will prevent the Broker from executing.

Database recovery procedures depend partly on the operating system on which the Broker is executing. This page describes the procedures for each operating system.

IBM i	The Universal Broker subsystem, **UNVUBR510** (by default), must be down in order to perform database recovery. Use standard IBM i database recovery procedures and attempt to restart the Universal Broker subsystem. If the problem persists, restore the failing database file. The entire Universal Spool file library may be required if restoring individual files fails to correct the problem. As a last resort, delete all files in the Universal Spool file library and restart **UNVUBR510**. Deleting the files from the Universal Spool library will result in loss of all data stored in those files, including spooled output for Manager Fault Tolerant jobs. All affected jobs may need to be re-run.

UNIX	The Broker daemon must be down to perform database recovery. A backup of either the database file being recovered or the entire directory should be created before recovery is attempted. A sample database recovery script is provided in file **ubrdbrec** in the `/opt/universal/ubroker/bin` directory. The script uses the Universal Database Utilities to dump and reload a database file. The default location of all Universal Broker databases is the `/var/opt/universal/spool` directory. The user ID with which the recovery script runs requires appropriate permissions to the database directory and to the database file. Write access is required to the directory and read and write access is required to the database file. The **ubrdbrec** script accepts an optional argument: the database file name to recover. If no database file name is specified, the **ues.db** database is recovered. The script ends with exit code 0 if successful and a non-zero exit code if it failed.
Windows	The Broker service must be stopped to perform database recovery. A backup of either the database file being recovered or the entire directory should be created before recovery is attempted. A sample database recovery batch file is provided in file **ubrdbrec.bat** in the `"\Program Files\Universal\UBroker\bin"` directory. The batch file uses the Universal Database Utilities to dump and reload a database file. The default location of all Universal Broker databases is directory `"\Program Files\Universal\spool\ubroker"`. The user ID with which the recovery script runs requires appropriate permissions to the database directory and to the database file. Write access is required to the directory and read and write access is required to the database file. The **ubrdbrec.bat** batch file accepts an optional argument: the database file name to recover. If no database file name is specified, the **ues.db** database is recovered. The batch file ends with exit code 0 if successful and a non-zero exit code if it failed.
z/OS	The Universal Broker started task must be down to perform database recovery. A backup of either the database file being recovered or the entire HFS or zFS data set should be created before recovery is attempted. A sample database recovery job is provided in member **UBRDBREC** in the **SUNVSAMP** library. The job uses the Universal Database Utilities to dump and reload a database file. All databases are located in the HFS or zFS product data set **#HLQ.UNV.UNVDB**. The HFS or zFS data set must be mounted prior to running **UBRDBREC**. See Mounting and Unmounting the Databases for information on mounting the HFS or zFS data set, if necessary. The user ID with which the recovery job runs requires appropriate permissions to the root directory of the HFS or zFS data set and to the database file. Write access is required to the directory and read and write access is required to the database file. Customize **UBRDBREC** to meet local JCL and installation requirements. Specify the database file name to recover on the PARM keyword of the **EXEC** statement of both steps (the dump and load steps). When all modifications are complete, submit the job. All steps should end with return code 0.

3.9 Database Recovery for Universal Enterprise Controller

3.9.1 Database Recovery for Universal Enterprise Controller

If Universal Enterprise Controller (UEC) terminates abnormally, it creates the file **uec.hf** in the database directory, which prompts UEC to initiate database verification upon restart.

Upon start-up, if UEC determines that an abnormal termination occurred, a verification process is performed on the database files. Verification tests the integrity of the files and determines if they are suitable for opening. If errors are detected and the integrity of the file is compromised, UEC reports the errors to the console and UEC immediately shuts down.

The Universal Database Dump (**UDBDUMP**) utility and the Universal Database Load (**UDBLOAD**) utility enable recovery from a corrupted Berkeley database.

Database recovery procedures depend partly on the operating system on which UEC is executing: z/OS or Windows. This page describes the procedures for each operating system.

Windows	The UEC service must be stopped to perform database recovery. A backup of either the database file being recovered or the entire directory should be created before recovery is attempted. A sample database recovery batch file is provided in file uecdbrec.bat in the `"\Program Files\Universal\UECtlr\bin"` directory. The batch file uses the Universal Database Utilities to dump and reload a database file. The default location of all UEC databases is `"\Program Files\Universal\UECtlr"`. **Note** Stonebranch has identified an issue with upgrades *from* releases earlier than UEC 3.2.0.0 (such as 3.1.0.x or 3.1.1.x) *to* releases 3.2.0.0 and later. Following the upgrade, UEC databases reside in the location specified by the user's currently configured working_directory location. This location defaults to "`\Program Files\Universal\UECtlr\bin`". If the current UEC install was not an upgrade, it may be necessary to pass the path to the **uec_evm.db** file as a command line argument to the script. You can provide an absolute path or a path relative to the **uecdbrec.bat** script's location. The user ID with which the recovery script runs requires appropriate permissions to the database directory and to the database file. Write access is required to the directory and read and write access is required to the database file. The **uecdbrec.bat** batch file accepts an optional argument-the database file name to recover. If no database file name is specified, the **uec_evm.db** database is recovered. The batch file ends with exit code 0 if successful and a non-zero exit code if it failed.
z/OS	The UEC started task must be down to perform database recovery. A backup of either the database file being recovered or the entire HFS or zFS data set should be created before recovery is attempted. A sample database recovery job is provided in member **UECDBREC** in the **SUNVSAMP** library. The job uses the Universal Database Utilities to dump and reload a database file. All databases are located in the HFS or zFS product data set **#HLQ.UNV.UECDB**. The HFS data set is allocated to the **UNVDB** ddname in both the dump and load steps. The HFS or zFS data set must be mounted prior to running **UECDBREC**. See Mounting and Unmounting the Databases for additional information on mounting the HFS or zFS data set. The user ID with which the recovery job runs requires appropriate permissions to the root directory of the HFS data set and to the database file. Write access is required to the directory and read and write access is required to the database file. Customize **UECDBREC** to meet local JCL and installation requirements. All UEC databases are recovered by the job. When all modifications are complete, submit the job. All steps should end with return code 0.

3.10 Listing Universal Agent Database Records Examples

3.10.1 Examples

- List Universal Broker Database
- List Universal Command Server Database Records
- List Universal Broker Detail for a Component
- List Standard Out for a Component

3.10.2 List Universal Broker Database

3.10.2.1 List Universal Broker Database

The following figures illustrate how to execute Universal Spool List (USLIST) with all defaults. (No options are required to issue USLIST.)

UNIX	`cd /opt/universal/bin` `uslist`
Windows	`cd /opt/universal/bin` `uslist`

Since no USLIST options are supplied, this example, by default, lists the contents of the Universal Broker Component database (UBROKER). A summary of all records is produced; no detail component records are written.

The broker database must be located in the default directory.

3.10.2.1.1 Components

Universal Spool List

3.10.3 List Universal Command Server Database Records

3.10.3.1 List Universal Command Server Database Records

The following figures illustrate how to list the database records for a specific component (in this case, Universal Command Server).

UNIX	`cd /opt/universal/bin` `uslist -list ucmd -ucmdspooldir "c:\program files\universal\spool2"`
Windows	`cd c:\program files\universal\uspool\bin` `uslist -list ucmd -ucmdspooldir "c:\program files\universal\spool2"`

These examples list the contents of the Universal Command Server Component database. A summary of all records is written.

These examples specify the directory location in which the Universal Command Server Component database is located. If the directory contains spaces, it must be enclosed in double (") quotes.

3.10.3.1.1 Command Line Options

The command line options used in this example are:

Option	Description
-list	Type of database from which to select record to write.
-ucmdspooldir	Directory location in which the Universal Command Server Component database (**scomponent.db**) is located.

3.10.3.1.2 Components

Universal Spool List

3.10.4 List Universal Broker Detail for a Component

3.10.4.1 List Universal Broker Detail for a Component

The following figures illustrate how to list the Universal Broker detail for a specific component ID.

UNIX	`cd /opt/universal/bin` `uslist -component 123456789`

Windows	`cd c:\program files\universal\uspool\bin` `uslist -component 123456789`

Since the **-list** option is not supplied, these examples, by default, list the contents of the Universal Broker Component database (UBROKER).

Because a component ID is specified, this will cause detail broker records to be written for component ID **123456789**.

3.10.4.1.1 Command Line Options

The command line option used in this example is:

Option	Description
-component	Component identifier for which records will be selected to write.

3.10.4.1.2 Components

Universal Spool List

3.10.5 List Standard Out for a Component

3.10.5.1 List Standard Out for a Component

The following figures illustrate how to list the standard output spool file for a specific component ID.

UNIX	`cd /opt/universal/bin` `uslist -list STDOUT -component 123456789`
Windows	`cd c:\program files\universal\uspool\bin` `uslist -list STDOUT -component 123456789`

The standard output spool file is written for component **123456789**.

3.10.5.1.1 Command Line Options

The command line options used in this example are:

Option	Description
-list	Type of database from which to select records to write.
-component	Component identifier for which records will be selected to write.

3.10.5.1.2 Components

Universal Spool List

3.11 Removing Universal Agent Database Records Examples

3.11.1 Examples

- Remove Component Records
- Remove Component Records - Change Database Directory

3.11.2 Remove Component Records

3.11.2.1 Remove Component Records

The following figures illustrate how to execute Universal Spool Remove (USLRM) with defaults.

UNIX	`cd /opt/universal/bin` `uslrm -component 123456789`
Windows	`cd c:\program files\universal\uspoolbin` `uslrm -component 123456789`

The only required option is **-component** (the component ID; you can execute Universal Spool List (USLIST) utility to find specific component IDs.

All Universal Agent database records will be removed for component **123456789**.

3.11.2.1.1 Command Line Options

The command line options used in this example are:

Command Options	Description
-component	Component identifier for which records will be removed.

3.11.2.1.2 Components

Universal Spool Remove

3.11.3 Remove Component Records - Change Database Directory

3.11.3.1 Remove Component Records - Change Broker Database Directory

The following figure illustrate how to execute Universal Spool Remove (USLRM) and specify a Universal Broker database directory other than the default.

UNIX	`cd /opt/universal/bin` `uslrm -component 123456789 -brokerspooldir "c:\program files\universal\spool2"`
Windows	`cd c:\program files\universal\uspool\bin` `uslrm -component 123456789 -brokerspooldir "c:\program files\universal\spool2"`

All Universal Agent database records will be removed.

The **-brokerspooldir** option specifies the directory location in which the Universal Broker Component database is located. If the directory has spaces, it must be enclosed within double (") quotation marks.

3.11.3.1.1 Command Line Options

The command line options used in this example are:

Command Options	Description
-component	Component identifier for which records will be removed.
-brokerspooldir	Directory location in which the Universal Broker Component database (**bcomponent.db**) is located.

3.11.3.1.2 Components

Universal Spool Remove

4 Universal Agent Security

4.1 Universal Agent Security

The following pages provide detailed information for Universal Agent Security:

- Security of Universal Agent Components
- Universal Access Control List (UACL)
- UACL Examples
- X.509 Certificates
- Creating Certificates Examples

4.2 Security of Universal Agent Components

4.2.1 Security of Universal Agent Components

Each component of Universal Agent is designed to be a secure system.

As the level of security rises, so does the administrative complexity of the system. Universal Agent balances the two, minimizing administrative complexity without sacrificing security.

These pages identify the security features inherent in the design for each Universal Agent component:

- Universal Broker
- Universal Command Manager
- Universal Command Server
- Universal Data Mover Manager
- Universal Data Mover Server
- Universal Event Monitor Manager
- Universal Event Monitor Server
- Universal Control Manager
- Universal Control Server
- Universal Event Log Dump
- Universal Spool List
- Universal Spool Remove

4.2.2 Universal Broker Security

- Overview
- File Permissions
- Configuration Files
- Universal Access Control List
- Universal Broker User Account

- Removing *ALLOBJ Authority from UNVUBR510 User Profile
- Removing *SPLCTL Authority from UNVUBR510 User Profile
- Removing *ALLOBJ and *SPLCTL Authorities from UNVUBR510 User Profile

4.2.2.1 Overview

Universal Broker is designed to be a secure system. As the level of security rises, so does the administrative complexity of the system. Universal Broker has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

Universal Broker security concerns are:

1. Access to Universal Broker files, directories, and configuration options.
2. Account with which Universal Broker executes.
3. Privacy and integrity of transmitted network data.

4.2.2.2 File Permissions

At a minimum, only trusted user accounts should have write access to the Universal Broker installation data sets. This most likely means only the administrators should have write access. For maximum security, only trusted accounts should have read access to these data sets.

IBM i	At a minimum, limit non-trusted user accounts to object authority of use to the Universal Broker product library, **UNVPRD510**; the product temporary library, **UNVTMP510**; the command reference library, **UNVCMDREF**; the universal spool library, **UNVSPL510**; and all objects within these libraries. For maximum security, only trusted accounts (administrators and the **UNVUBR510** profile) should have management, existence, alter, add, update, or delete authority to these objects. As a reminder, the system value **QCRTAUT** controls public access authority to created objects unless overridden by specific commands. User profiles that use Stonebranch products require *CHANGE authority to the **UNVTMP510** library.
HP NonStop	All files that the Broker creates or updates are located in either **$SYSTEM.UNVLOG** or **$SYSTEM.UNVTRACE**. The Broker does not need write access to its installation subvolume.
UNIX	All files that the Broker creates or updates are located in the **/var/opt/universal** directory. This means that the Broker does not need write access to its installation directory or subdirectories. Universal Broker requires full control (read, write, remove, and add) of the **/var/opt/universal** directory and its subdirectories. The Broker creates spool files, trace files, and log files in this directory. Only the account used to execute the Broker requires full access to this directory. The Broker configuration options can be changed to use directories other then **/var/opt/universal**. If this is the case, the same permissions must be set up for these specified directories.
Windows	Universal Broker requires write access to its primary install directory (that is, **.\Universal\UBroker**), which serves as its default trace file location. The Broker requires full control over the database directory (that is, **.\Universal\spool**), along with all subdirectories and files under that location. When the Broker executes as a console application with its message destination set to **logfile**, the Broker requires full control over the **.\Universal\Broker\log** directory and all .log files within it. The Universal Broker Windows service always writes its message to the Windows event log, which means that it requires no write access to a log directory or any other of its installation subdirectories and files.

z/OS	Universal Broker requires update access to its database files, which exist as HFS- or zFS-allocated datasets mounted on the z/OS Unix System Services (z/OS USS) file system. The Broker accesses HFS-allocated datasets using the **UNVDB** and **UNVSPOOL** ddnames in its STC JCL. The Broker accesses zFS-allocated datasets via its UNIX_DB_DATA_SET and UNIX_SPOOL_DATA_SET configuration options.

4.2.2.3 Configuration Files

Only trusted user accounts should have write, create or delete access to the Broker configuration files or any of the directories in the configuration file directory search list.

Windows	Although you can edit Universal Agent configuration files with any text editor (for example, Notepad), we recommend using the Universal Configuration Manager Control Panel application set configuration options. The Universal Configuration Manager provides a graphical interface and context-sensitive help, and helps protect the integrity of the configuration file by validating all changes to configuration option values. It also directs the Broker to refresh its cache of Universal Agent component configuration settings, making it unnecessary to issue a separate configuration REFRESH request via the Universal Control utility.

4.2.2.4 Universal Access Control List

Universal Broker uses the Universal Access Control List (UACL) as an extra layer of security. The UACL contains entries (that is, rules) that permit or deny access to the Universal Broker (see Universal Access Control List (UACL) for details).

Universal Broker reads the UACL entries when the program is started. If the UACL file is changed, the new entries can be activated either by:

- Stopping and starting Universal Broker.
- Sending Universal Broker a Universal Control configuration refresh request, which instructs Universal Broker to reread all of its configuration files, including the UACL file (see Configuration Refresh).

Windows	Although you can edit the UACL file with any text editor (for example, Notepad), we recommend that you maintain UACL entries using the Universal Configuration Manager Control Panel application. The Universal Configuration Manager sends a configuration refresh request to the Universal Broker. Updated values take effect immediately, making it unnecessary to recycle the Broker to apply UACL changes.

4.2.2.5 Universal Broker User Account

Each Universal Agent component that the Universal Broker spawns inherits the Broker's account credentials. Occasionally, components must perform privileged operations, such as establishing a process execution environment using a local user account's credentials.

On some platforms, this means that the Broker must execute with an account whose inherited credentials allow the spawned components to perform these operations. On other platforms, the Broker may be executed with a lesser-privileged user, provided that the components are configured in way that permits them to elevate their privileges when necessary.

The section contains platform-specific requirements to consider when setting the Broker's user account.

IBM i

Universal Broker for IBM i runs with the **UNVUBR510** user profile, which is created at product installation time. Any component started by Universal Broker inherits this user profile.

By default, the **UNVUBR510** user profile has *ALLOBJ, *JOBCTL, and *SPLCTL authority. Unless the user profile is modified as described in the following section, *ALLOBJ authority is required for a component to switch its user profiles based on the request it is servicing. *JOBCTL authority is required for internal control and should not be removed. The **UNVUBR510** user profile requires *SPLCTL authority to provide Universal Submit Job job logs in specific, limited situations.

Any other product or user should not use the **UNVUBR510** user profile. By default, users cannot access the system with the **UNVUBR510** profile.

Removing *ALLOBJ Authority from UNVUBR510 User Profile

Given the extensive authority allowed by *ALLOBJ special authority, it is desirable to avoid its use when possible. As of PTF 0UC0126 for V1R2M1, it is possible to remove *ALLOBJ special authority from the **UNVUBR510** user profile. However, by removing *ALLOBJ from the **UNVUBR510** user profile, the administrative complexity is increased.

The following steps are required to use Universal Command with *ALLOBJ special authority removed from the **UNVUBR510** user profile.

1. If the following objects do not have *USE Public Authority, the **UNVUBR510** user profile must be given *USE authority:

- QSYS/QSYGETPH
- QSYS/QWTSETP
- QSYS/QWCRJBST
- QSYS/QUSRMBRD

This can be accomplished with the following command:

```
===> EDTOBJAUT OBJ(QSYS/object_name) OBJTYPE(*PGM)
```

From the resulting screen, use F6 to add user **UNVUBR510** and give it *USE authority.

2. **UNVUBR510** user profile must be given *USE authority to the user profile objects of all user profiles that will be using the universal command server on the IBM i.

This can be accomplished with the following command:

```
===> EDTOBJAUT OBJ(QSYS/user_profile_name) OBJTYPE(*USRPRF)
```

From the resulting screen, use F6 to add user **UNVUBR510** and give it *USE authority.

3. Use the following command to remove the **UNVUBR510** user profile *ALLOBJ authority:

```
===> CHGUSRPRF USRPRF(UNVUBR510) SPCAUT(*JOBCTL *SPLCTL)
```

Removing *SPLCTL Authority from UNVUBR510 User Profile

Use the following command to remove the **UNVUBR510** user profile *SPLCTL authority:

```
===> CHGUSRPRF USRPRF(UNVUBR510) SPCAUT(*JOBCTL *ALLOBJ)
```

Removing *ALLOBJ and *SPLCTL Authorities from UNVUBR510 User Profile

Use the following command to remove all special authority from the **UNVUBR510** user profile:

	`` ===> CHGUSRPRF USRPRF(UNVUBR510) SPCAUT(*JOBCTL) ``` (Please refer to the previous two sections for additional information.)
HP NonStop	Universal Broker itself does not require **super.super** privileges. For example, Universal Command (UCMD) Server may require **super.super** authority. Since the component inherits its user ID from the Broker, either the Broker must be running as **super.super** or the UCMD Server program must be owned by **super.super** and **ProgID** must be set for the server program file. If the Broker is started as a daemon at system startup time, it is started with a user ID of super.super. The Broker and all its components will then have sufficient authority.
UNIX	Although Universal Broker itself does not require superuser privileges, some Universal Agent server components (for example, UCMD Server and UEM Server) may require superuser authority to switch execution context to another user account, initialize group membership, or perform other privileged operations. Since the component inherits its user ID from Universal Broker, one of the following is required: • Universal Broker must execute as **root**. • **root** must own the Universal Agent Server application file (for example, **ucmsrv** or **uemsrv**), and the Universal Agent Server application file must have its "set user ID on execution" bit (**setuid on exec**) set (for example, **chmod u+s ucmsrv**). By default, the Universal Broker is owned and started with a user ID of **ubroker**. **root** will own the server components that need superuser authority and these components will have their "set user ID on execution" bit set. **Note** Universal Agent server components typically only invoke the privileged operations mentioned above when that component is configured to run with security enabled (that is, its security configuration option is set to a value other than **none**). When security is disabled in a Universal Agent server component's configuration, that component may not attempt to invoke any privileged operations, but relies completely upon the security context it inherits from the Broker.
Windows	The Universal Broker Windows service can be configured to execute with the Local System account or with a specially-configured Administrative account (see Windows Service).
z/OS	The Universal Broker started task may execute with any OMVS user ID provided that account has read access to the **BPX.DAEMON**, **BPX.SUPERUSER**, and **BPX.JOBNAME** resources in the FACILITY class. The Broker user account is typically configured at install time (see Started Task Configuration). **Note** • Starting with Universal Broker 5.1.0.1, the Broker USER ID no longer requires READ access to the BPX.SUPERUSER resource. • Starting with Universal Broker 6.5.0.0, the Broker USER ID no longer requires READ access to the BPX.DAEMON resource. See z/OS Configuration - Started Tasks for more information.

4.2.3 Universal Command Manager Security

4.2.3.1 File Permissions

Only trusted user accounts should have write permission to the Universal Command (UCMD) Manager installation directory, its subdirectories, and all files within those directories. This most likely means only an administration group should have write access.

IBM i	Only trusted user accounts and the **UNVUBR520** user profile should have write permission to the UCMD Server product library (**UNVPRD510**) and all files within that library. Eligible users of UCMD require read access to the message catalogs in the **UNVNLS** file.
HP NonStop	Eligible users of UCMD require read access to the message catalogs in the **$SYSTEM.UNVNLS** subvolume.
UNIX	Eligible users of UCMD require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the Universal Agent installation directory (**/opt/universal** by default).
Windows	Eligible users of UCMD require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the **.\Universal** installation directory. These file permissions are set automatically during the installation.
z/OS	Only trusted user accounts should have write access to the UCMD Manager installation files. Eligible users of UCMD require read access to the national language support library (**SUNVNLS**), the configuration file (**UNVCONF**), and the load library (**SUNVLOAD**).

4.2.3.2 RACF Protection

z/OS	The UCMD Manager for z/OS verifies a users access to a RACF general resource profile. The resource profile controls a user's access to execute a command on a remote host with a specific remote user identity. See the z/OS Installation - Configuration of Security for complete details on installing and administering UCMD Manager RACF profiles.

4.2.3.3 Configuration Files

Only trusted user accounts should have write access to the UCMD Manager configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.4 Universal Command Server Security

4.2.4.1 File Permissions

Only trusted user accounts should have write permission to the Universal Command (UCMD) Server installation directory, subdirectories, and all files within them.

IBM i	Only administrator accounts should have write permission to the UCMD Server product library, **UNVPRD510**; the command reference library, **UNVCMDREF**; the universal spool library, **UNVSPL510** and all objects within these libraries. For maximum security, only trusted accounts (administrators and the **UNVUBR510** profile) should have management, existence, alter, add, update, and delete authority to these objects. As a reminder, the system value **QCRTAUT** controls public access authority to created objects unless overridden by specific commands. Other than users authorized to use Universal Agent components, the same applies to the product temporary library, **UNVTMP510**.
Windows	Only trusted user accounts should have write permission to the UCMD Server installation directory and subdirectories, and all of the files within them. This most likely means only the administrator group should have write access. Eligible users of Universal Command require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the Universal Agent installation directory. All eligible users of UCMD require permission to create directories in the UCMD Server working directory, if security is activated. A directory named after the user ID requesting the command is created for each user. The directory is created while impersonating the user; hence, it's created using the user's security account. Home directories are created with permissions giving the user full control of both the directory and the files within the directory.
z/OS	Only trusted user accounts should have write permission to the UCMD Server installation data sets. No general user access is required.

4.2.4.2 Configuration Files

Only trusted user accounts should have write access to the UCMD Server configuration files.

Windows	Although you may edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group may execute the Universal Configuration Manager.

4.2.4.3 Universal Command Server User ID

UNIX	UCMD Server requires read access to its installation directory and its working directory (defined in the component definition). If user security is activated, the Server requires root access to create processes that execute with another user's identity. The Server security identity is inherited from the Broker. If the Broker is running with a non-root user ID, then the Server program must have the set user ID on execution permission set and root as owner. See Universal Message Translator for details.
z/OS	UCMD Server for z/OS requires read access to its installation data sets and its HFS working directory (defined in the component definition).

4.2.4.4 Universal Command Server User Profile

IBM i	If user security is activated, the UCMD Server for IBM i requires, by default, *ALLOBJ authority to change user profiles. Unless modifications are made (as described in **Removing *ALLOBJ Authority from UNVUBR510 User Profile** in the IBM i section of **Universal Broker User Account** in Universal Broker Security), the Server user profile, which is inherited from the Broker, requires *ALLOBJ authority.

4.2.4.5 User Authentication

User authentication is the process of verifying that a user is a known and valid user. The process used by UCMD Server requires the user to provide an operating system-specific user name / ID and a password. The UCMD Server passes the name / ID and password to the operating system for verification; this is referred to as logging on the user.

Windows	Windows provides two primary types of log on processes: batch and interactive. A user must be given the right to log on as a batch job for them to do a batch log on. All users can do an interactive log on. See the LOGON_METHOD option for more details.
UNIX	Universal Command can use three different types of user authentication methods: 1. Default authentication uses the UNIX traditional password comparison method. 2. PAM authentication uses the PAM API to authenticate users and, optionally, process session modules. This option is available only for certain UNIX platforms. 3. HP-UX Trusted Security uses HP-UX Trust Security APIs to authenticate users. This is available only on Hewlett Packard HP-UX and Tru64 platforms.

HP-UX 11.00 and later	By default, supplemental group memberships are recorded in the **/etc/group** file. However, if an **/etc/logingroup** file exists, it governs all supplemental group memberships and effectively overrides the entries in **/etc/group**. **Note** **/etc/logingroup** is not required to record supplemental group membership. If **/etc/logingroup** does not exist, **/etc/group** is sufficient to record the groups in which a user belongs. If any Universal Agent component fails to access system resources that are secured based on supplemental group membership, make sure that the authenticated user has an entry in **/etc/logingroup**, if that file exists. Otherwise, the default entry in **/etc/group** should be sufficient. For more information about **/etc/logingroup**, please see the HP-UX system documentation.
IBM i	If the user name and password are successfully validated by the operating system, the Initiator program (**UCMSINIT**) changes the current user profile to the user profile of the user ID.

4.2.4.6 Command References

A command reference is file, residing on a Universal Command (UCMD) Server system, which contains a pre-defined command or script to be executed upon request of a Universal Command Manager.

When used with Universal Access Control List (UACL) entries, command references allow UCMD administrators to restrict what commands and processes can be executed by remote UCMD Managers.

For more information, see Universal Command - Command References.

4.2.5 Universal Data Mover Manager Security

4.2.5.1 Universal Data Mover Manager Security

Universal Data Mover is designed to be a secure system. As the level of security rises, so does the administrative complexity of the system. Universal Data Mover has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

Universal Data Mover security concerns are:

1. Access to Universal Data Mover files and directories
2. Access to Universal Data Mover configuration files
3. Universal Data Mover user account
4. Privacy and integrity of transmitted network data

5. User authentication

4.2.5.2 File Permissions

Only trusted user accounts should have permission to write to the Universal Data Mover installation directory and subdirectories, and all files within those directories.

IBM i	**Object Permissions** Only administrator accounts should have write permission to the following Universal Agent libraries (and all objects within these libraries): • Installation library, UNVPRD510 (by default) • Product temporary library, UNVTMP510 • Universal spool library, UNVSPL510 For maximum security, only trusted accounts (administrators and the UNVUBR510 profile) should have management, existence, alter, add, update, and delete authority to these objects. **Note** System value QCRTAUT controls public access authority to created objects unless overridden by specific commands.
z/OS	**Data Set Permissions** Only trusted user accounts should have write access to the Universal Data Mover installation files. Eligible users of Universal Data Mover require read access to the national language support library SUNVNLS, the configuration file UNVCONF, and the load library SUNVLOAD.

4.2.5.3 Configuration Files

Only trusted user accounts should have write access to the Universal Data Mover Manager configuration files.

Windows	Although you may edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.6 Universal Data Mover Server Security

4.2.6.1 Universal Data Mover Server Security

Universal Data Mover Server is designed to be a secure system. As the level of security rises, so does the administrative complexity of the system. Universal Data Mover Server has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

Universal Data Mover Server security concerns are:

- Access to product data sets
- Access to Universal Agent configuration files
- Universal Broker user account
- Privacy and integrity of transmitted network data
- User authentication

4.2.6.2 File Permissions

Only trusted user accounts should have write permission to the Universal Data Mover Server installation directory and sub-directories, and all of the files within them.

IBM i	**Object Permissions** Only administrator accounts should have write permission to the following Universal Agent libraries (and all objects within these libraries): • Installation library, UNVPRD510 (by default) • Product temporary library, UNVTMP510 • Universal spool library, UNVSPL510 For maximum security, only trusted accounts (administrators and the UNVUBR510 user profile) should have management, existence, alter, add, update or delete authority to these objects. As a reminder, the system value QCRTAUT controls public access authority to created objects unless overridden by specific commands.
z/OS	**Data Set Permissions** Only trusted user accounts should have write permission to the Universal Data Mover Server installation data sets. No general user access is required.

4.2.6.3 Configuration Files

Only trusted user accounts should have write access to the Universal Data Mover Server configuration files.

Windows	Although you may edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.6.4 Universal Data Mover Server User ID

Universal Data Mover Server requires read access to its installation directory and its working directory (defined in the component definition).

UNIX	If user security is activated, the Server requires root access to create processes that execute with another user's identity. The Server security identity is inherited from the Broker. If the Broker is running with a non-root user ID, then the Server program must have the set user ID on execution permission set and root as owner.
z/OS	Universal Data Mover Server requires read access to its installation data sets and its HFS working directory (defined in the component definition).

4.2.6.5 Universal Data Mover Server User Profile

IBM i	If user security is activated, the UDM Server for IBM i requires, by default, *ALLOBJ authority to switch user profiles. This *ALLOBJ authority requirement may be removed. The UDM Server initially inherits authority from the UNVUBR510 user profile. Following the switch to the user profile, the UDM Server runs under the authority of the user initiating the data transfer. The UNVUBR510 user profile requires *SPLCTL authority in order to provide Universal Submit Job with job logs in specific limited situations. The *SPLCTL authority requirement can be removed. Removing *SPLCTL from the UNVUBR510 user profile may prevent the job log processing in limited situations. (See Universal Broker User Account for information on removing the *ALLOBJ and *SPLCTL authorities.)

4.2.6.6 User Authentication

User authentication is the process of verifying that a user is known and valid to the system. The process used by UDM Server requires the user to provide a user name / ID and a password. The UDM Server passes the name / ID and password to the operating system for verification; this is referred to as logging on the user.

IBM i	For IBM i, user authentication is optional. However, if security is enabled, a user name / ID and password are required in order to verify the user's credentials. With security enabled, you transfer files using a specific user's security context.
UNIX	For UNIX, user authentication is optional. However, if security is enabled, a user name / ID and password are required in order to verify the user's credentials. With security enabled, you transfer files using a specific user's security context. Universal Data Mover can use three different types of user authentication methods: 1. Default authentication uses the UNIX traditional password comparison method. 2. PAM authentication uses the PAM API to authenticate users. The PAM modules, which authenticate and account, are called. This option is available only for certain UNIX platforms. 3. HP-UX Trusted Security uses HP-UX Trust Security APIs to authenticate users. This is available only on Hewlett Packard HP-UX platforms.

HP-UX 11.00 and later	By default, supplemental group memberships are recorded in the **/etc/group** file. However, if an **/etc/logingroup** file exists, it governs all supplemental group memberships and effectively overrides the entries in **/etc/group**. **Note** **/etc/logingroup** is not required to record supplemental group membership. If **/etc/logingroup** does not exist, **/etc/group** is sufficient to record the groups in which a user belongs. If any Universal Agent component fails to access system resources that are secured based on supplemental group membership, make sure that the authenticated user has an entry in **/etc/logingroup**, if that file exists. Otherwise, the default entry in **/etc/group** should be sufficient. For more information about **/etc/logingroup**, please see the HP-UX system documentation.
Windows	For Windows, user authentication is optional. However, if security is enabled, a user name / ID and password are required in order to verify the user's credentials. (With security enabled, you transfer files using a specific user's security context.)

4.2.7 Universal Event Monitor Manager Security

4.2.7.1 File Permissions

Only trusted user accounts, which are most likely those that are members of the Administrators group, should be granted write access to the UEM Manager installation directory and subdirectories, and the files within them.

UNIX	Authorized users of UEM require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the primary Universal Agent installation directory.
Windows	Authorized users of UEM require read access to the message catalogs (***.umc** files), which reside in the **.\Universal\nls** directory. If UEM Manager is installed on an NTFS partition, these file permissions are set automatically during the installation.
z/OS	Eligible users of UEM require read access to the national language support library **SUNVNLS**, the configuration file **UNVCONF**, and the load library **SUNVLOAD**.

4.2.7.2 Data Privacy

Data transmitted from a UEM Manager across a network connection to the Universal Broker and demand-driven UEM Server is protected using features present in all Stonebranch Inc. Universal Agent components.

For more information on the steps taken to protect transferred data, see Network Data Transmission.

4.2.7.3 RACF Protection

z/OS	The UEM Manager for z/OS verifies a user's access to a RACF general resource profile. The resource profile controls a user's ability to monitor an event on a remote host with a specific remote user identity. See the Configuration of Security for complete details on installing and administering UEM Manager RACF profiles.

4.2.7.4 Configuration Files

Only trusted user accounts should have write access to the Universal Event Monitor Manager configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.8 Universal Event Monitor Server Security

4.2.8.1 Data Privacy

Data transmitted to a UEM Server across a network connection is protected using features present in all Stonebranch Inc. Universal Agent components.

For more information on the steps taken to protected transferred data, see Network Data Transmission.

4.2.8.2 File Permissions

Only trusted user accounts should have write access to the UEM Server installation directory and sub-directories, and the files within them. Authorized users of UEM require read access to the message catalogs (***.umc** files), which reside in the **./universal/nls** directory.

Windows	If UEM Server is installed on an NTFS partition, these file permissions are automatically set during installation. The component definitions for demand-driven and event-driven UEM Servers include the location of a WORKING_DIRECTORY. By default, this is **.\Universal\UEMHome**. When the USER_SECURITY option is enabled, and before a demand-driven UEM Server begins monitoring an event or an event-driven UEM Server executes an event handler process, the UEM Server will create a subdirectory (if it does not already exist) for the authenticated user under this working directory. The name of the directory matches the ID of the user account specified from the UEM Manager command line or stored in the event handler record. If a Windows domain account is used, the name of the directory is **userid.domain**, where **userid** is the user ID and **domain** is the domain name. After the directory is created, the specified user account is given ownership of it and granted full control over it.

4.2.8.3 Configuration Files

Only trusted user accounts should have write access to the Universal Event Monitor Server configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.8.4 User Authentication

UNIX	When the USER_SECURITY option is enabled, a demand-driven UEM Server requires the ID of a valid local user account before it will begin monitoring the event. A password also may be required, depending on the rules set up in ACCESS_ACL. Likewise, an event-driven UEM Server requires this information to be stored in an event handler record before it can execute a process on behalf of that handler. All handler processes started by UEM Server when the USER_SECURITY option is enabled are executed in the security context of this user account. UEM Server for UNIX supports three different types of user authentication methods: 1. Default authentication uses the UNIX traditional password comparison method. 2. PAM authentication uses the PAM API to authenticate users. This option is only available for certain UNIX platforms. 3. HP-UX Trusted Security uses HP-UX Trust Security APIs to authenticate users. This is only available on Hewlett Packard HP-UX platforms.

HP-UX 11.00 and later	By default, supplemental group memberships are recorded in the **/etc/group** file. However, if an **/etc/logingroup** file exists, it governs all supplemental group memberships and effectively overrides the entries in **/etc/group**. **Note** **/etc/logingroup** is not required to record supplemental group membership. If **/etc/logingroup** does not exist, **/etc/group** is sufficient to record the groups in which a user belongs. If any Universal Agent component fails to access system resources that are secured based on supplemental group membership, make sure that the authenticated user has an entry in **/etc/logingroup**, if that file exists. Otherwise, the default entry in **/etc/group** should be sufficient. For more information about **/etc/logingroup**, please see the HP-UX system documentation.
Windows	When the USER_SECURITY option is enabled, a demand-driven UEM Server requires the ID and password of a valid local user account before it will begin monitoring the event. Likewise, an event-driven UEM Server requires this information to be stored in an event handler record before it can execute a process on behalf of that handler. All handler processes started by UEM Server when the USER_SECURITY option is enabled are executed in the security context of this user account. To allow Windows to verify the user account information, a UEM Server will attempt to log that user on to the system via a call to a Windows system function. Windows provides two types of logon methods: interactive and batch. Unless they have been explicitly denied the ability to do so, most user accounts can be validated with the interactive logon method. Conversely, a user account typically must be granted an additional privilege before they can be authenticated using the batch logon method. This privilege is shown in Windows as "Log on as a batch job." For information on configuring UEM Server to use this logon method, see the UEM Server LOGON_METHOD option.

4.2.9 Universal Control Manager Security

4.2.9.1 File Permissions

Only trusted user accounts should have write permission to the Universal Control Manager installation directory and subdirectories, and all of the files within them. This most likely means that only the administrator group should have write access.

Eligible users of Universal Control require read access to the message catalogs (***.umc** files) in the NLS directory.

HP NonStop	Eligible users of Universal Control require read access to the message catalogs in the **$SYSTEM.UNVNLS** subvolume.

Windows	Eligible users of Universal Control require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the Universal Agent installation directory. If Universal Control Manager is installed on an **NTFS** partition, these file permissions are set automatically during the installation.
z/OS	Data set permissions: Eligible users of Universal Control require read access to: • National language support library **SUNVNLS** • Load library **SUNVLOAD**

4.2.9.2 Configuration Files

Only trusted user accounts should have write access to the Universal Control Manager configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.9.3 RACF Protection

z/OS	The Universal Control Manager for z/OS verifies a user's access to a RACF general resource profile. The resource profile controls a user's access to execute a control request on a remote host. See the Configuration of Security for complete details on installing and administering Universal Control Manager RACF profiles.

4.2.10 Universal Control Server Security

4.2.10.1 File Permissions

Only trusted user accounts should have write permission to the Universal Control Server installation directory and subdirectories, and all of the files within them.

HP NonStop	Object permissions: Only trusted user accounts should have management, existence, alter, add, update or delete authority to the Universal Control Server installation libraries and objects.

Windows	Eligible users of UCTL require read access to the message catalogs (***.umc** files) in the **nls** subdirectory of the Universal Agent installation directory. If security is activated, all eligible users of UCTL require permission to create directories in the UCTL Server working directory. A directory named after the user ID requesting the command is created for each user. The directory is created while impersonating the user; hence, it is created using the user's security account. Home directories are created with permissions giving the user full control of both the directory and the files within them.

4.2.10.2 Configuration Files

Only trusted user accounts should have write access to the Universal Control Server configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.10.3 Universal Control Server User ID

Universal Control Server requires read access to its installation directory and its working directory (defined in the component definition). The Universal Control Server security identity is inherited from the Universal Broker.

IBM i	The associated user profile (**UNVUBR510**) provides *ALLOBJ authority.
z/OS	UCTL Server requires read access to its installation data sets and its HFS working directory (defined in the component definition).

4.2.10.4 User Authentication

User authentication is the process of verifying that a user is a known and valid user. The process used by Universal Control Server requires the user to provide a user name / ID and a password. The Universal Control Server passes the name / ID and password to the operating system for verification; this is referred to as logging on the user.

Windows	Windows provides two primary types of log on processes: batch and interactive. A user must be given the right to log on as a batch job in order for the user to do a batch log on. All users can do an interactive log on. (See the Universal Control Server LOGON_METHOD option for more details.)

4.2.11 Universal Event Log Dump Security

- Security Access
- Event Log Access

- Universal Configuration Manager

4.2.11.1 Security Access

No special security access is required to run Universal Event Log Dump (UELD). However, accessing the event logs and setting configuration options may require some special security considerations.

4.2.11.2 Event Log Access

The system and application event logs may be read by all user accounts. The security log can only be accessed by accounts with Administrator privileges. Administrator privileges are also required to clear any of the event logs.

4.2.11.3 Universal Configuration Manager

Only trusted user accounts should have write access to the Universal Event Log Dump configuration files.

Windows	Although you can edit configuration files with any text editor (for example, Notepad), we recommend that you manage configuration options using the Universal Configuration Manager Control Panel application. Only user accounts in the Administrator group can execute the Universal Configuration Manager.

4.2.12 Universal Spool List Security

4.2.12.1 Account Access

The account used to execute the Universal Spool List utility must have read access to the database files listed in Universal Agent Databases.

4.2.13 Universal Spool Remove Security

4.2.13.1 Account Access

The user account used to run the Universal Spool Remove utility must have read/write access to the database files listed in Universal Agent Databases.

4.3 Universal Access Control List (UACL)

- Overview
- UACL Configuration

4.3.1 Overview

Many Universal Agent components utilize the Universal Access Control List (UACL) feature as an extra layer of security to the services they offer. UACLs are used for a variety of reasons but generally are used to determine if a client request is allowed or denied permission to the service and to set security attributes for the client request.

Each Universal Broker has an associated UACL configuration file that contains all the UACL entries for that system. The UACL entries can be used to enforce a security policy specific to the system on which its deployed.

The following Universal Agent components use the UACL feature:

Component	Description
Universal Automation Center Agent	UACLs are used to control whether user credentials are required for task execution and to control whether or not user authentication is required. See Universal Automation Center Agent UACL Entries for complete details.
Universal Broker	UACLs are used to permit or deny TCP/IP client connections. See Universal Broker UACL Entries for complete details.
Universal Command Server	UACLs are used to permit or deny Universal Command Manager access and to control whether or not the Manager request requires user authentication. See Universal Command UACL for complete details.
Universal Control Server	UACLs are used to permit or deny Universal Control Manager access and to control whether or not the Manager request requires user authentication. See Universal Control UACL Entries for complete details.
Universal Data Mover Server	UACLs are used to permit or deny Universal Data Manager access and to control whether or not the Manager request requires user authentication. See Universal Data Mover UACL Entries for complete details.
Universal Event Monitor Server	UACLs are used to permit or deny Universal Event Monitor Manager access and to control user authentication for event handlers. See Universal Event Monitor UACL Entries for complete details.
Universal Message Service	UACLs are used to permit or deny TCP/IP client connections and provide access to the OMS Administration Utility. See OMS Server UACL Entries for complete details.

Note

For component-specific examples of UACL entries, see UACL Examples.

4.3.2 UACL Configuration

UACL entries are maintained in a configuration file. The UACL configuration file is required for the Universal Broker to start even if there are no UACL entries defined in it.

The UACL configuration file syntax is the same as all other Universal Agent configuration files except for one difference: multiple UACL entries of the same name may be defined. The order in which the UACL entries are listed in the configuration file determines the order in which they are searched. See Configuration File Syntax for details on configuration file syntax.

The following table describes the location of the UACL configuration file and how it is accessed for each platform.

Platform	Description
z/OS	All UACL entries are defined in member **ACLCFG00** in library **UNVCONF**. The Universal Broker started task allocates the UACL configuration file to ddname **UNVACL**.
UNIX	All UACL entries are defined in the **uacl.conf** configuration file. This file is installed in `/etc/universal` by default. The UACL file is searched for in the same manner as all other product configuration files.
Windows	All UACL entries are defined in the **uacl.conf** configuration file. The location of this file depends on the version of Windows. It is recommended to use the Windows Universal Configuration Manager to view and update UACL entries.
IBM i	All UACL entries are defined in member **UACL** of file **UNVCONF**.

4.3.3 UACL Entries

4.3.3.1 UACL Entries

UACL entries are specified in the UACL configuration file. All Universal Agent UACL configuration files are simple text files. UACL entries are defined one per line in the following format:

```
TYPE RULE
```

- `TYPE` identifies the UACL entry. Each Universal Agent component using UACL entries have a defined set of entry types it supports. For example, the Universal Broker component uses UACL entries of type **ubroker_access**.
- `RULE` defines the UACL entry matching criteria, the client access, and potentially some additional security attributes. The client's identity and the client's request are used to match UACL rules

UACL entries of the same type are listed in the order in which they should be searched. Since all UACL entries are in the same UACL configuration file, it is highly recommended to keep entries grouped together by their type for easy maintenance.

There is no limit to the number of UACL entries that can be specified in the UACL configuration file.

4.3.3.1.1 UACL Entries Example

An example of UACL entries in a UACL configuration file is listed below.

```
ucmd_access         10.20.30.,TS1004,tsup1004,allow,noauth
ucmd_access         10.20.30.,TS1004,*,allow,auth
ucmd_access         10.20.30.,*,*,deny,auth
ucmd_access         ALL,*,root,deny,auth

ucmd_cert_access    joe,tsup1004,allow,noauth
ucmd_cert_access    joe,*,allow,auth
ucmd_cert_access    operations,*,deny,auth
ucmd_cert_access    *,root,deny,auth
```

4.3.3.1.2 Rule Syntax

The UACL entry rule consists of a comma-separated list of values. If there is a space or tab character in the list of values, the entire list must be enclosed in quotation (") characters. What values are required and the meaning of each value is specific to the UACL entry type and are defined in the Universal Agent component documentation for its UACL entries.

As an example, the following **ucmd_request** UACL entry contains a space in the "DSPLIB QGPL" rule value, so the entire rule is enclosed in quotation characters.

```
ucmd_request   "prod.host.name,remoteuser,localuser,cmd,DSPLIB QGPL,allow,auth"
```

4.3.3.1.3 Generics

You can use generics - pattern control characters and codes - in the UACL rule to match client requests.

Generics allow you to specify a string pattern in the rule to match a client request value. A string pattern is a convenient way of specifying one or more values.

4.3.3.1.3.1 Pattern Control Characters

The following pattern control characters can be used:

Control Character	Description
*	Match 0 or more characters.
?	Match one character.
/	Escape character to escape matching control characters so they are used as literal characters and to specify control codes.

4.3.3.1.3.2 Pattern Control Codes

In addition to the pattern control characters, pattern control codes can be specified to control how the pattern matching is performed. Pattern control codes are specified in the pattern string by prefixing them with the escape character, which is the slash character (/).

The following pattern control codes can be used:

Control Codes	Description
c	Perform a case insensitive compare.
C	Perform a case sensitive compare (the default).
s	Normalize spaces by reducing multiple spaces to one.
S	Don't normalize spaces (the default).

4.3.3.1.3.3 Example String Patterns Using Generics

Some example string patterns using generics are listed below.

Pattern	Description
*le	matches "apple", "le", and "red apple".
/*le	matches "*le" only.
ap?le	matches "apple", "ap le", but not "aple".
/c*le	matches "apple", "APPLE", and "appLe".
a/c*le	matches "apple", "aPpLe", but not "APPLE".
/s*le	matches "apple", "red apple", and "red apple".

4.3.3.1.4 Host Name Aliases

For any UACL entry that accepts the IP address or host name of a client, you can specify a canonical name for the host (that is, an alias) instead of the actual host name. This allows you to update DNS CNAME records to point to new hosts without having to update UACL entries that apply to that host, as might happen when switching from a test to production environment.

To indicate that the host name specified in the UACL rule is (or could be) an alias, prefix the name with the @ symbol.

When the @ symbol is used and the application is unable to match the IP address and host name of the client host to the UACL entry, the application tries to obtain information about the host name that follows the @ symbol. If that host name value is an alias of the client host, the application examines the other UACL entry parameters. Otherwise, the application skips the UACL entry and proceeds to the next one.

For example, the following ubroker_access entry indicates that `myalias.abc.com` could be an alias for a remote client:

```
ubroker_access @myalias.abc.com,allow
```

The application applies this rule if `myalias.abc.com` is the actual host name, or an alias for it, of the client. Without the @ symbol, the application ignores the rule if the actual host name of the client is something else (for example, `myhost.abc.com`).

Although using alias names provides a measure of flexibility with respect to Agent configurations, such entries should be used with caution. While some entries may be examined only at process start-up, the expense of a round-trip query to the host name resolver (for example, DNS) - particularly when multiplied across several processes - may have an undesirable effect on overall workflow performance. You may want to adopt alias name usage gradually, starting with your most dynamic environment, in order to measure any performance impacts that occur.

4.3.3.2 Searching UACL Entries

UACL entries are searched in the order they are listed in the UACL configuration file. The search criteria is based on the client identity and the client request. Once a matching UACL entry has been found, the search stops and the matching entry is used.

The client identity is defined as a combination of the client TCP/IP IP address, client TCP/IP host name, client user ID, and client digital certificate. See Client Identification for details on client identification.

The client request is defined based on the UACL entry type. There is typically an UACL entry type representing the different types of client requests. Each rule value has fields that correspond to the client request values. See Request Identification for details on request identification.

4.3.3.3 Additional Information

The following pages provide additional detailed information for UACL Entries:

- Client Identification
- Request Identification
- Certificate- and Non Certificate-Based Entries
- UACL Entries for Universal Agent Components

4.3.3.4 Client Identification

4.3.3.4.1 Client Identification Methods

Rule matching is based on the client identity and the client request.

There are two client identification methods:

1. X.509 Certificate Authentication
2. Client IP Address Identification

4.3.3.4.2 X.509 Certificate Authentication

X.509 certificates identify an entity. An entity can be a program, person, or host computer. When an X.509 certificate is authenticated, it authenticates that the entity is who it claims to be.

X.509 certificates are utilized in UACL entries by first mapping a client certificate to a UACL certificate identifier. The certificate identifier then is used in the UACL entries. A certificate identifier provides for:

1. Concise representation of certificates in UACL entries. There are a large number of certificate fields that may be used and many of the fields have lengthy, tedious naming formats. A certificate map only needs to be defined once and then the concise certificate identifier can be used in the UACL entries.
2. Mapping of one or more certificates to a single certificate identity. A group of entities that share a common security access level may be represented by one certificate identity reducing the number of UACL entries to maintain.

UACL certificate map entries are searched sequentially (that is, top to bottom) matching the client certificate to each entry until a match is found. The certificate map defines a set of X.509 certificate fields that may be used as matching criteria.

4.3.3.4.2.1 Certificate Map Matching Criteria

The following table defines the certificate map matching criteria.

Criteria	Description
SUBJECT	Matches the X.509 subject field. The subject field is formatted as an X.501 Distinguished Name (DN). A DN is a hierarchical list of attributes referred to as Relative Distinguished Names (RDNs). RDNs are separated with a comma (,) by default. If a different separator is required (perhaps one of the RDN values uses a comma), start the DN with the different separator character. Valid separators are slash (/), comma (,), semicolon (;), and period (.). Many RDN values can be used in a DN. Some of the most common values are: • C (Country name) • CN (Common name) • L (Locality) • O (Organization) • OU (Organizational Unit) • ST (State) The RDN attributes must be list The RDN attributes must be listed in the same order as they are defined in the certificate to be considered matched. A partial DN can be specified. All certificates that have a subject name that matches up to the last RDN are considered a match. This permits a group of certificates to be matched. The RDN attribute values can include pattern matching characters. An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. Some example of SUBJECT values are: • **subject="C=US,ST=Georgia,O=Acme,CN=Road Runner"** • **subject="C=US,ST=Georgia,O=Acme,CN=Road * "** • **subject="C=US,ST=Georgia,O=Acme,CN=Road ?unner"** Whether or not an RDN value is case sensitive depends on the format in which the value is stored. The certificate creator has some control over which format is used. All formats except for printableString are case sensitive.

Criteria	Description
EMAIL	Matches the X.509 **emailAddress** attribute of the **subject** field and **rfc822Name** of the **subjectAltName** extension value. Both fields format the email address as an RFC 822 **addr-spec** in the form of **identifier@domain**. The attribute values may include pattern matching characters. An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. Some example EMAIL values are: • **email=user1@acme.com** • **email=*@acme.com** • **email=user?@acme.com** RFC 822 names are not case sensitive.
HOSTNAME	Matches the following X.509 fields in the order listed: 1. **dNSName** of the **subjectAltName** extension value. 2. **commonName** (CN) RDN attribute of the **subject** field's DN value. Some example HOSTNAME values are: • **hostname=bigfish.acme.com** • **hostname=*.acme.com** The values are not case sensitive.
IPADDRESS	Matches the X.509 **iPAddress** field of the **subjectAltName** extension value. An example IPADDRESS value is: • **ipaddress=10.20.30.40**
SERIALNUMBER	Matches the X.509 **serialNumber** value. The value can be specified in a hexadecimal format by prefixing the value with **0x** or **0X**, otherwise, the value is considered a decimal format. For example, the value **0x016A392E7F** would be considered a hexadecimal format. An example SERIALNUMBER value is: • **serialnumber=0x7a2d52cbae**

4.3.3.4.2.2 Certificate Identifier Field

If a certificate map rule is found that matches the client certificate, the rule's identifier is assigned to the client's request. The certificate identifier is then used in matching certificate-based UACL entries.

The following table defines the certificate identifier field as used in UACL entries.

Field	Description
CERTID	Matches the certificate identifier defined by the certificate map entry. The CERTID value has the following syntax: • An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. For example, **AB*M** matches **ABCDM** and **ABM**. **AB?M** matches **ABCM**, but not **ABCDM**. • The comparison is case insensitive. • Pattern matching characters, such as the asterisk and question mark, are included in the text to be matched by prefixing them with a forward slash (/) character. For example, **A/*B** matches **A*B**. **A//B** matches **A/B**.

4.3.3.4.3 Client IP Address Identification

TCP/IP provides a method to obtain a client's IP address. The IP address typically identifies the host computer on which the client is executing. However, there are exceptions to this. Networks can be configured with Network Address Translation (NAT) systems between the client and the Broker that hides the client's IP address. In addition to the client IP address, Universal Agent clients provide a user account name with which they are executing that is used to further refine the client's identity.

UACL entries are searched matching the client's IP address and user account to each entry until a match is found.

4.3.3.4.3.1 Client IP Address - Matching Criteria

The following table defines possible matching criteria for IP address and user account client identification.

Field	Description
HOST	Matches the TCP/IP address of the remote user. The HOST value has the following syntax: • Dotted numeric form of an IP address. For example, **10.20.30.40**. • Dotted numeric prefix of the IP addresses. For example, **10.20.30.** matches all IP addresses starting with **10.20.30**. The last dot (.) is required. • A net/mask expression. For example, **131.155.72.0/255.255.254.0** matches IP address range **131.155.72.0** through **131.155.76.255**. The mask and the host value are AND'ed together. The result must match net. **Note** Contact your network administrator for calculation of the correct net / mask expression. • Host name for an IP address. For example, **sysa.abc.com**. • Host name suffix for a range of IP addresses. For example, **.abc.com** matches all host names ending with **abc.com**, such as, **sysa.abc.com**. The first dot (.) is required. • An alias of a defined host. The application recognizes values prefixed with the **@** symbol as a possible alias to the actual host name of a client. For example, an entry of **@sysa.xyz.com** matches a client whose actual host name is **sysa.abc.com**, provided that **sysa.xyz.com** is a defined alias of that host. Note that the entry is also considered a match if **sysa.xyz.com** is the actual host name of the client. • A value of **ALL** matches all IP addresses. The value must be uppercase.
REMOTE_USER	Matches the user name with which the remote user is executing as on the remote system. The REMOTE_USER value has the following syntax: • An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. For example, **AB*M** matches **ABCDM** and **ABM**. **AB?M** matches **ABCM** but not **ABCDM**. • Control code /c switches off case-sensitivity and /C switches on case?sensitivity matching. The default is on. For example, **/cABC** matches **abc**. **/ca/Cbc** matches **Abc** but not **ABC**. • Pattern matching characters, such as the asterisk and question mark, are included in the text to be matched by prefixing them with a forward slash (/) character. For example, **A/*B** matches **A*B**. **A//B** matches **A/B**.

4.3.3.5 Request Identification

4.3.3.5.1 Request Identification

In addition to the client identity being used to search for UACL entries, the client's request may be part of the matching criteria. The exact request fields used are dependent on the component's UACL entry type.

4.3.3.5.1.1 Request Fields

The following table lists a complete set of possible request fields. See each component's UACL entry definitions for further details.

Field	Description
LOCAL_USER	Matches the local user name with which the remote user is requesting to execute as on the local host. LOCAL_USER value has the following syntax: • An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. For example, *AB*M* matches *ABCDM* and *ABM*. *AB?M* matches *ABCM* but not *ABCDM*. • Control code /c switches off case\-sensitivity and /C switches on case?sensitivity matching. The default is on. For example, */cABC* matches *abc*. */ca/Cbc* matches *Abc* but not *ABC*. • Pattern matching characters, such as the asterisk and question mark, are included in the text to be matched by prefixing them with a forward slash (/) character. For example, *A/*B* matches *A*B*. *A//B* matches *A/B*. • Variable name *$RMTUSER* can be included in the value. The variable name itself is not case sensitive. *$RMTUSER* and *$rmtuser* are the same. The *$RMTUSER* variable value is the user name with which the remote user is executing. It is the same value used in matching the REMOTE_USER field. A space character delimits the variable name, or it can be enclosed in parentheses (for example, **$(RMTUSER)**), in which case it is delimited by the right parenthesis. This is useful if it is immediately followed by text. For example, if the remote user name is **TOM**, a LOCAL_USER value of **$RMTUSER** will match if the local user name requested is also **TOM**. A LOCAL_USER value of **$(RMTUSER)01** will match if the local user name requested is **TOM01**. **Windows** The LOCAL_USER value is not case sensitive, since Windows user account names are not case sensitive.
REQUEST_TYPE	Matches the type of request a Universal Command Manager is requesting. The REQUEST_TYPE value has the following syntax: • An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. For example, **AB*M** matches **ABCDM** and **ABM**. **AB?M** matches **ABCM** but not **ABCDM**. • The comparison is case insensitive. • Pattern matching characters, such as the asterisk and question mark, are included in the text to be matched by prefixing them with a forward slash (/) character. For example, **A/*B** matches **A*B**. **A//B** matches **A/B**.

Field	Description
REQUEST_NAME	The REQUEST_NAME field matches the name of a Universal Command Manager is request. The REQUEST_NAME value has the following syntax: • An asterisk (*) matches 0 or more characters and a question mark (?) matches one character. For example, **AB*M** matches **ABCDM** and **ABM**. **AB?M** matches **ABCM** but not **ABCDM**. • Case sensitivity depends on the REQUEST_TYPE and the operating system on which the Universal Command Server is executing. See the Server's Security section for the operating system in question. • Control code **/c** switches off case-sensitivity and **/C** switches on case?sensitivity matching. The default is on. For example, **/cABC** matches **abc**. **/ca/Cbc** matches **Abc** but not **ABC**. • Control code **/s** normalizes spaces and **/S** does not normalize spaces. Space normalization removes preceding and trailing spaces as well as reduce consecutive multiple spaces to a single space. The default is no space normalization. For example, **/sa b c** matches **a b c**. **/Sa b c** matches **a b c** but not **a bc**. • Pattern matching characters, such as the asterisk and question mark, are included in the text to be matched by prefixing them with a forward slash (/) character. For example, **A/*B** matches **A*B**. **A//B** matches **A/B**.

4.3.3.6 Certificate- and Non Certificate-Based Entries

4.3.3.6.1 Certificate-Based and Non Certificate-Based UACL Entries

Universal Agent components that support X.509 certificates define their UACL entries in two varieties:

1. Certificate-based entries
2. Non certificate-based entries

The two entry types are distinguished by their name. For example, **ucmd_cert_access** is the certificate-based form of the entry and **ucmd_access** is a non certificate-based entry. All entries follow the same format.

Either the certificate-based UACL entries or the non certificate-based UACL entries are searched, but not both.

4.3.3.6.1.1 Certificate-Based Entries Search

Certificate-based UACL entries are searched under the following conditions:

- Client provides an X.509 certificate that matches a certificate map entry.

4.3.3.6.1.2 Non Certificate-Based Entries Search

Non certificate-based UACL entries are searched under the following conditions:

- Client provides an X.509 certificate and no certificate map entry matches.
- Client does not provide an X.509 certificate.

4.3.3.7 UACL Entries for Universal Agent Components

4.3.3.7.1 UACL Entries for Universal Agent Components

The following UACL entries are available for Universal Agent components:

- Universal Broker UACL Entries
- Universal Automation Center Agent UACL Entries
- Universal Command UACL Entries
- Universal Data Mover UACL Entries
- Universal Event Monitor UACL Entries
- Universal Control UACL Entries

4.3.4 UACL Examples

4.3.4.1 UACL Examples

- Universal Broker for zOS - UACL Example
- Universal Broker for Windows - UACL Example
- Universal Broker for UNIX - UACL Example
- Universal Broker for IBM i - UACL Example
- Universal Command Server for zOS - UACL Example
- Universal Command Server for Windows - UACL Example
- Universal Command Server for UNIX - UACL Example
- Universal Command Server for IBM i - UACL Example
- Universal Control Server for zOS - UACL Example
- Universal Control Server for Windows - UACL Example
- Universal Control Server for UNIX - UACL Example
- Universal Control Server for IBM i - UACL Example
- Universal Data Mover Server for zOS - UACL Example
- Universal Data Mover Server for Windows - UACL Example
- Universal Data Mover Server for UNIX - UACL Example
- Universal Data Mover Server for IBM i - UACL Example
- Universal Event Monitor Server for Windows - UACL Example
- Universal Event Monitor Server for UNIX - UACL Examplev

4.3.4.2 Universal Broker for zOS - UACL Example

4.3.4.2.1 Universal Broker for z/OS

The following set of rules authorize the Universal Enterprise Controller at address 10.20.30, with update access to the product configuration files and setting of the configuration managed mode of the Broker, and denies all other connections.

```
remote_config_access     10.20.30.,*,allow,allow
remote_config_access     ALL,*,deny,deny
```

The following set of rules permit connections for the subnet 10.20.30 and denies all other connections.

```
ubroker_access     10.20.30.,allow
ubroker_access     ALL,deny
```

The following set of rules permit connections from host 10.20.30.40 and 10.20.30.50 and denies all other connections.

```
ubroker_access     10.20.30.40,allow
ubroker_access     10.20.30.50,allow
ubroker_access     ALL,deny
```

The following set of rules map X.509 certificates to certificate identifiers.

```
cert_map           id=joe,subject="/C=US/ST=Georgia/O=Acme, Inc./
                                  OU=Sales/CN=Joe Black"
```

4.3.4.2.1.1 Components

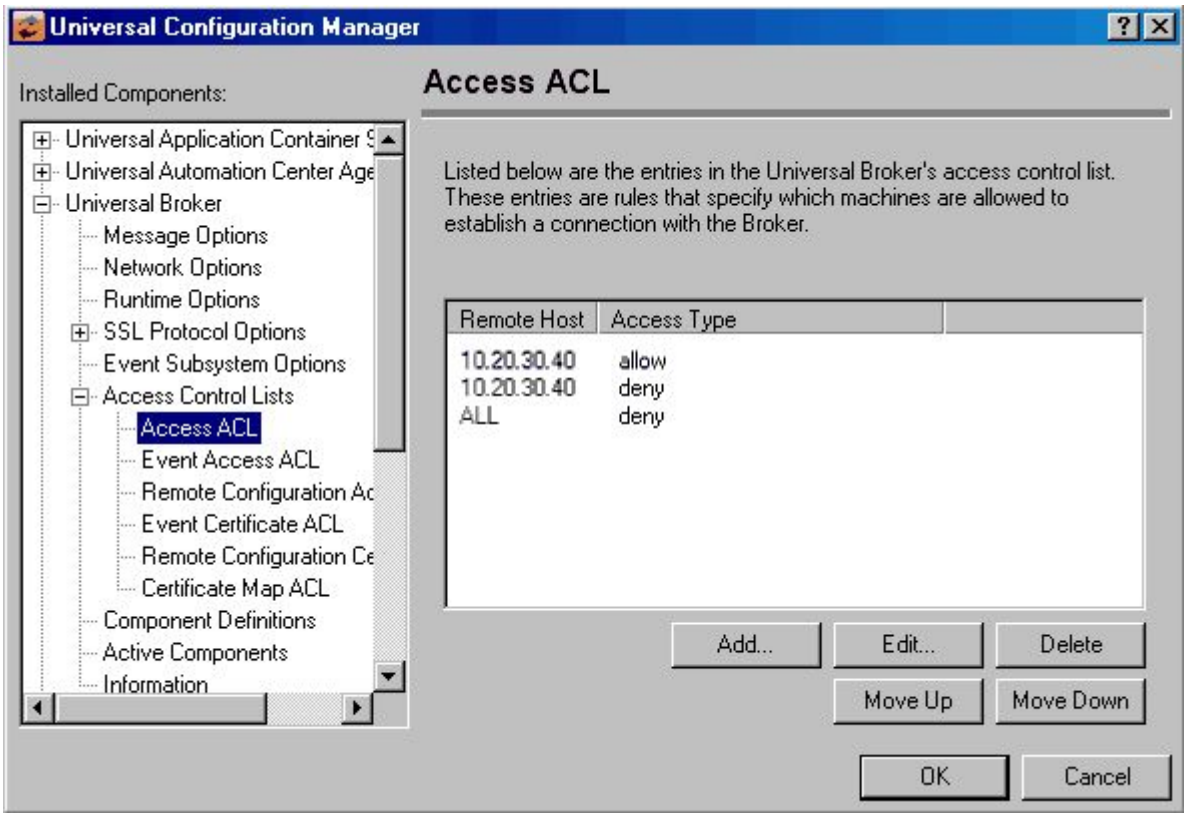
Universal Broker for z/OS

4.3.4.3 Universal Broker for Windows - UACL Example

4.3.4.3.1 Universal Broker for Windows

Although UACL files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to update UACL entries. From there, ACL entries can be added, changed, deleted or sorted (rules are applied in the order in which they are listed).

The following figure illustrates an example.



4.3.4.3.1.1 Components

Universal Broker for Windows

4.3.4.4 Universal Broker for UNIX - UACL Example

4.3.4.4.1 Universal Broker for UNIX

The following set of rules is required to allow I-Management Console to access Universal Broker.

```
remote_config_access     10.20.30.,*,allow,allow
remote-config_access     ALL,*,deny,deny
```

The following set of rules permit connections for the subnet 10.20.30 and denies all other connections.

```
ubroker_access     10.20.30.,allow
ubroker_access     ALL,deny
```

The following set of rules permit connections from host 10.20.30.40 and 10.20.30.50 and denies all other connections.

```
ubroker_access     10.20.30.40,allow
ubroker_access     10.20.30.50,allow
ubroker_access     ALL,deny
```

The following set of rules map X.509 certificates to certificate identifiers.

```
cert_map           id=joe,subject="/C=US/ST=Georgia/O=Acme, Inc./"
```

4.3.4.4.1.1 Components

Universal Broker for UNIX

4.3.4.5 Universal Broker for IBM i - UACL Example

4.3.4.5.1 Universal Broker for IBM i

The following set of rules permit connections for the subnet 10.20.30 and denies all other connections.

```
ubroker_access     10.20.30.,*,allow,allow
ubroker_access     ALL,*,deny,deny
```

The following set of rules permit connections from host 10.20.30.40 and 10.20.30.50 and denies all other connections.

```
ubroker_access     10.20.30.40,allow
ubroker_access     10.20.30.50,allow
ubroker_access     ALL,deny
```

The following set of rules map X.509 certificates to certificate identifiers.

```
cert_map           id=joe,subject="/C=US/ST=Georgia/O=Acme, Inc./
                                  OU=Sales/CN=Joe Black"
```

4.3.4.5.1.1 Components

Universal Broker for IBM i

4.3.4.6 Universal Command Server for zOS - UACL Example

4.3.4.6.1 Universal Command Server for z/OS

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
ucmd_access     10.20.30.,*,*,allow,auth
ucmd_access     ALL,*,*,deny,auth

ucmd_cert_access  operations,*,allow,auth
ucmd_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user **TS1004** on that host.

- No host can execute commands as local user **SUPERID**.
- User **TS1004** on host 10.20.30.40 can execute commands as local user **TSUP1004** without providing the password.
- Users **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID **TSUP1004** without a password.

- Certificate ID **joe** is allowed to execute commands with any other local user ID with a password.
- Certificate ID operations cannot run anything.
- All other certificate IDs can execute commands with any user ID except for **SUPERID** with a password.

```
ucmd_access     10.20.30.40,TS1004,tsup1004,allow,noauth
ucmd_access     10.20.30.40,TS1004,*,allow,auth
ucmd_access     10.20.30.40,*,*,deny,auth
ucmd_access     ALL,*,superid,deny,auth

ucmd_cert_access   joe,tsup1004,allow,noauth
ucmd_cert_access   joe,*,allow,auth
ucmd_cert_access   operations,*,deny,auth
ucmd_cert_access   *,superid,deny,auth
```

4.3.4.6.1.1 Components

Universal Command Server for z/OS

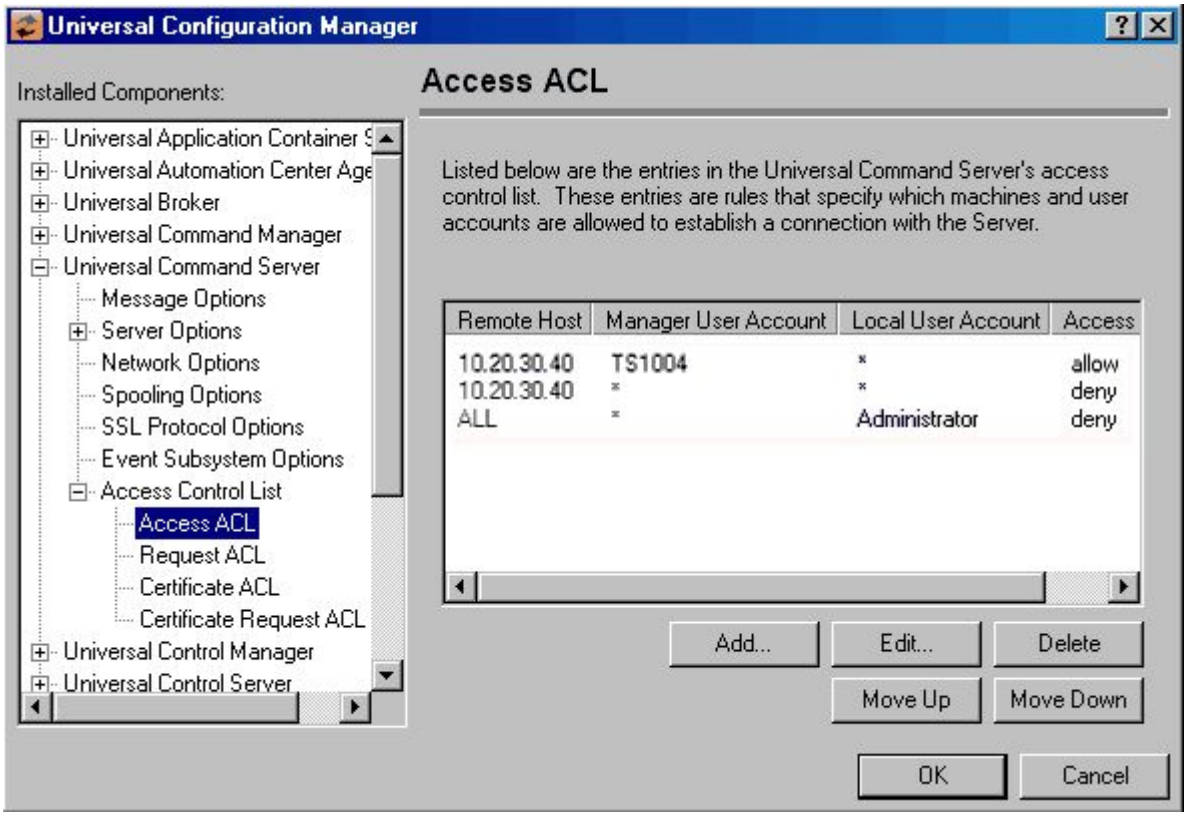
4.3.4.7 Universal Command Server for Windows - UACL Example

4.3.4.7.1 Universal Command Server for Windows

Although UACL files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to update UACL entries. From there, ACL entries can be added, changed, deleted or sorted (rules are applied in the order in which they are listed).

The following figure illustrates an example. The set of ACL entries only allows connections from host 10.20.30.40 if the user on that host is TS1004. All other remote users will be blocked.

- TS1004 may run processes on the local system using any user account, provided the correct password is supplied.
- No processes may be run with Universal Command using the Administrator account on the local system, regardless of where the request originated.



4.3.4.7.1.1 Components

Universal Command Server for Windows

4.3.4.8 Universal Command Server for UNIX - UACL Example

4.3.4.8.1 Universal Command Server for UNIX

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
ucmd_access     10.20.30.,*,*,allow,auth
ucmd_access     ALL,*,*,deny,auth

ucmd_cert_access  operations,*,allow,auth
ucmd_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can execute commands as local user root.
- User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID tsup1004 without a password.

- Certificate ID **joe** is allowed to execute commands with any other local user ID with a password.
- Certificate ID **operations** cannot run anything.
- All other certificate IDs can execute commands with any user ID except for root with a password.

```
ucmd_access     10.20.30.40,TS1004,tsup1004,allow,noauth
ucmd_access     10.20.30.40,TS1004,*,allow,auth
ucmd_access     10.20.30.40,*,*,deny,auth
ucmd_access     ALL,*,root,deny,auth

ucmd_cert_access   joe,tsup1004,allow,noauth
ucmd_cert_access   joe,*,allow,auth
ucmd_cert_access   operations,*,deny,auth
ucmd_cert_access   *,root,deny,auth
```

4.3.4.8.1.1 Components

Universal Command Server for UNIX

4.3.4.9 Universal Command Server for IBM i - UACL Example

4.3.4.9.1 Universal Command Server for IBM i

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
ucmd_access     10.20.30.,*,*,allow,auth
ucmd_access     ALL,*,*,deny,auth

ucmd_cert_access  operations,*,allow,auth
ucmd_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user **TS1004** on that host.

- No host can execute commands as local user **root**.
- User **TS1004** on host 10.20.30.40 can execute commands as local user **tsup1004** without providing the password.
- Users **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID **tsup1004** without a password.

- Certificate ID **joe** is allowed to execute commands with any other local user ID with a password.
- Certificate ID operations cannot run anything.
- All other certificate IDs can execute commands with any user ID except for root with a password.

```
ucmd_access     10.20.30.40,TS1004,tsup1004,allow,noauth
ucmd_access     10.20.30.40,TS1004,*,allow,auth
ucmd_access     10.20.30.40,*,*,deny,auth
ucmd_access     ALL,*,root,deny,auth

ucmd_cert_access   joe,tsup1004,allow,noauth
ucmd_cert_access   joe,*,allow,auth
ucmd_cert_access   operations,*,deny,auth
ucmd_cert_access   *,root,deny,auth
```

4.3.4.9.1.1 Components

Universal Command Server for IBM i

4.3.4.10 Universal Control Server for zOS - UACL Example

4.3.4.10.1 Universal Control Server for z/OS

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
uctl_access     10.20.30.,*,*,allow,auth
uctl_access     ALL,*,*,deny,auth

uctl_cert_access  operations,*,allow,auth
uctl_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can execute commands as local user root.
- User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user id TSUP1004 without a password.

- Certificate ID joe is allowed to execute commands with any other local user ID with a password.
- Certificate ID operations cannot run anything.
- All other certificate IDs can execute commands with any user ID except for SUPERID with a password.

```
uctl_access     10.20.30.40,TS1004,tsup1004,allow,noauth
uctl_access     10.20.30.40,TS1004,*,allow,auth
uctl_access     10.20.30.40,*,*,deny,auth
uctl_access     ALL,*,root,deny,auth

uctl_cert_access   joe,tsup1004,allow,noauth
uctl_cert_access   joe,*,allow,auth
uctl_cert_access   operations,*,deny,auth
uctl_cert_access   *,root,deny,auth
```

4.3.4.10.1.1 Components

Universal Control

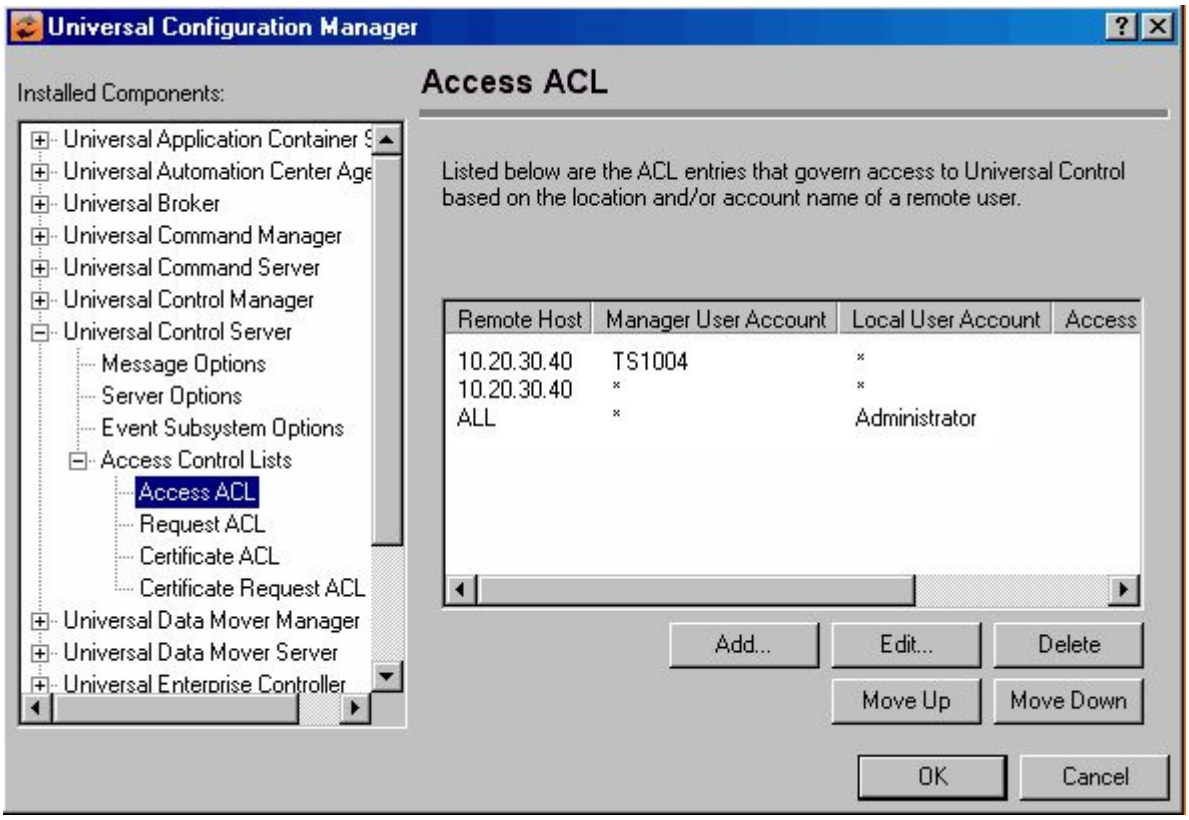
4.3.4.11 Universal Control Server for Windows - UACL Example

4.3.4.11.1 Universal Control Server for Windows

Although UACL files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to update UACL entries. From there, ACL entries can be added, changed, deleted or sorted (rules are applied in the order in which they are listed).

The following figure illustrates an example. The set of ACL entries only allows connections from host 10.20.30.40 if the user on that host is TS1004. All other remote users will be blocked.

- TS1004 may run processes on the local system using any user account, provided the correct password is supplied.
- No processes may be run with Universal Command using the Administrator account on the local system, regardless of where the request originated.



4.3.4.11.1.1 Components

Universal Control

4.3.4.12 Universal Control Server for UNIX - UACL Example

4.3.4.12.1 Universal Control Server for UNIX

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
uctl_access     10.20.30.,*,*,allow,auth
uctl_access     ALL,*,*,deny,auth

uctl_cert_access  operations,*,allow,auth
uctl_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permits connections from any host, but has limited access from host 10.20.30.40 to user **TS1004** on that host.

- No host can execute commands as local user **root**.
- User **TS1004** on host 10.20.30.40 can execute commands as local user **tsup1004** without providing the password.
- User **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID **joe** can request local user id t*sup1004* without a password.

- Certificate ID **joe** is allowed to execute commands with any other local user ID with a password.
- Certificate ID operations cannot run anything.
- All other certificate IDs can execute commands with any user ID except for root with a password.

```
uctl_access     10.20.30.40,TS1004,tsup1004,allow,noauth
uctl_access     10.20.30.40,TS1004,*,allow,auth
uctl_access     10.20.30.40,*,*,deny,auth
uctl_access     ALL,*,root,deny,auth

uctl_cert_access   joe,tsup1004,allow,noauth
uctl_cert_access   joe,*,allow,auth
uctl_cert_access   operations,*,deny,auth
uctl_cert_access   *,root,deny,auth
```

4.3.4.12.1.1 Components

Universal Control

4.3.4.13 Universal Control Server for IBM i - UACL Example

4.3.4.13.1 Universal Control Server for IBM i

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
uctl_access     10.20.30.,*,*,allow,auth
uctl_access     ALL,*,*,deny,auth

uctl_cert_access  operations,*,allow,auth
uctl_cert_access  *,*,deny,auth
```

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host but has limited access from host 10.20.30.40 to user **TS1004** on that host.

- No host can execute commands as local user root.
- User **TS1004** on host 10.20.30.40 can execute commands as local user **tsup1004** without providing the password.
- Users **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID **joe** can request local user ID **tsup1004** without a password.

- Certificate ID joe is allowed to execute commands with any other local user ID with a password.
- Certificate ID operations cannot run anything.
- All other certificate IDs can execute commands with any user ID except for root with a password.

```
uctl_access     10.20.30.40,TS1004,tsup1004,allow,noauth
uctl_access     10.20.30.40,TS1004,*,allow,auth
uctl_access     10.20.30.40,*,*,deny,auth
uctl_access     ALL,*,root,deny,auth

uctl_cert_access   joe,tsup1004,allow,noauth
uctl_cert_access   joe,*,allow,auth
uctl_cert_access   operations,*,deny,auth
```

4.3.4.13.1.1 Components

Universal Control

4.3.4.14 Universal Data Mover Server for zOS - UACL Example

4.3.4.14.1 Universal Data Mover Server for z/OS

The following set of rules permit services for the subnet 10.20.30 and denies all other connections.

```
udm_access      10.20.30.,*,*,allow,auth
udm_access      ALL,*,*,deny,auth
```

The following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can execute commands as local user root.
- User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

```
udm_access      10.20.30.40,TS1004,tsup1004,allow,noauth
udm_access      10.20.30.40,TS1004,*,allow,auth
udm_access      10.20.30.40,*,*,deny,auth
udm_access      ALL,*,root,deny,auth
```

4.3.4.14.1.1 Components

Universal Data Mover Server for z/OS

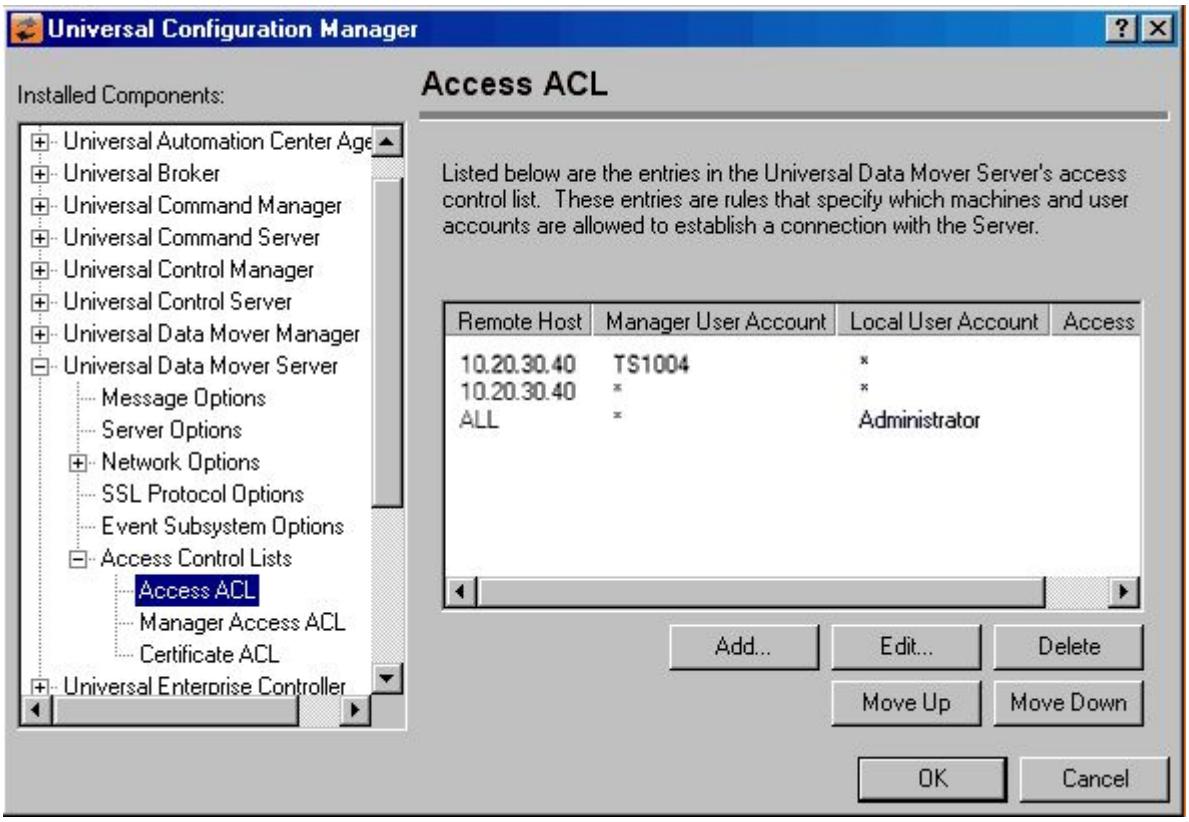
4.3.4.15 Universal Data Mover Server for Windows - UACL Example

4.3.4.15.1 Universal Data Mover Server for Windows

Although UACL files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to update UACL entries. From there, ACL entries can be added, changed, deleted or sorted (rules are applied in the order in which they are listed).

The following figure illustrates an example. The set of ACL entries only allows connections from host 10.20.30.40 if the user on that host is TS1004. All other remote users will be blocked.

- TS1004 may run processes on the local system using any user account, provided the correct password is supplied.
- No processes may be run with Universal Data Mover using the Administrator account on the local system, regardless of where the request originated.



4.3.4.15.1.1 Components

Universal Data Mover Server for Windows

4.3.4.16 Universal Data Mover Server for UNIX - UACL Example

4.3.4.16.1 Universal Data Mover Server for UNIX

The following set of rules permit services for the subnet 10.20.30 and denies all other connections.

```
udm_access      10.20.30.,*,*,allow,auth
udm_access      ALL,*,*,deny,auth
```

The following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can execute commands as local user root.
- User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

```
udm_access      10.20.30.40,TS1004,tsup1004,allow,noauth
udm_access      10.20.30.40,TS1004,*,allow,auth
udm_access      10.20.30.40,*,*,deny,auth
udm_access      ALL,*,root,deny,auth
```

4.3.4.16.1.1 Components

Universal Data Mover Server for UNIX

4.3.4.17 Universal Data Mover Server for IBM i - UACL Example

4.3.4.17.1 Universal Data Mover Server for IBM i

The following set of rules permit services for the subnet 10.20.30 and denies all other connections.

```
udm_access      10.20.30.,*,*,allow,auth
udm_access      ALL,*,*,deny,auth
```

The following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can execute commands as local user root.
- User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

```
udm_access      10.20.30.40,TS1004,tsup1004,allow,noauth
udm_access      10.20.30.40,TS1004,*,allow,auth
udm_access      10.20.30.40,*,*,deny,auth
udm_access      ALL,*,root,deny,auth
```

4.3.4.17.1.1 Components

Universal Data Mover Server for IBM i

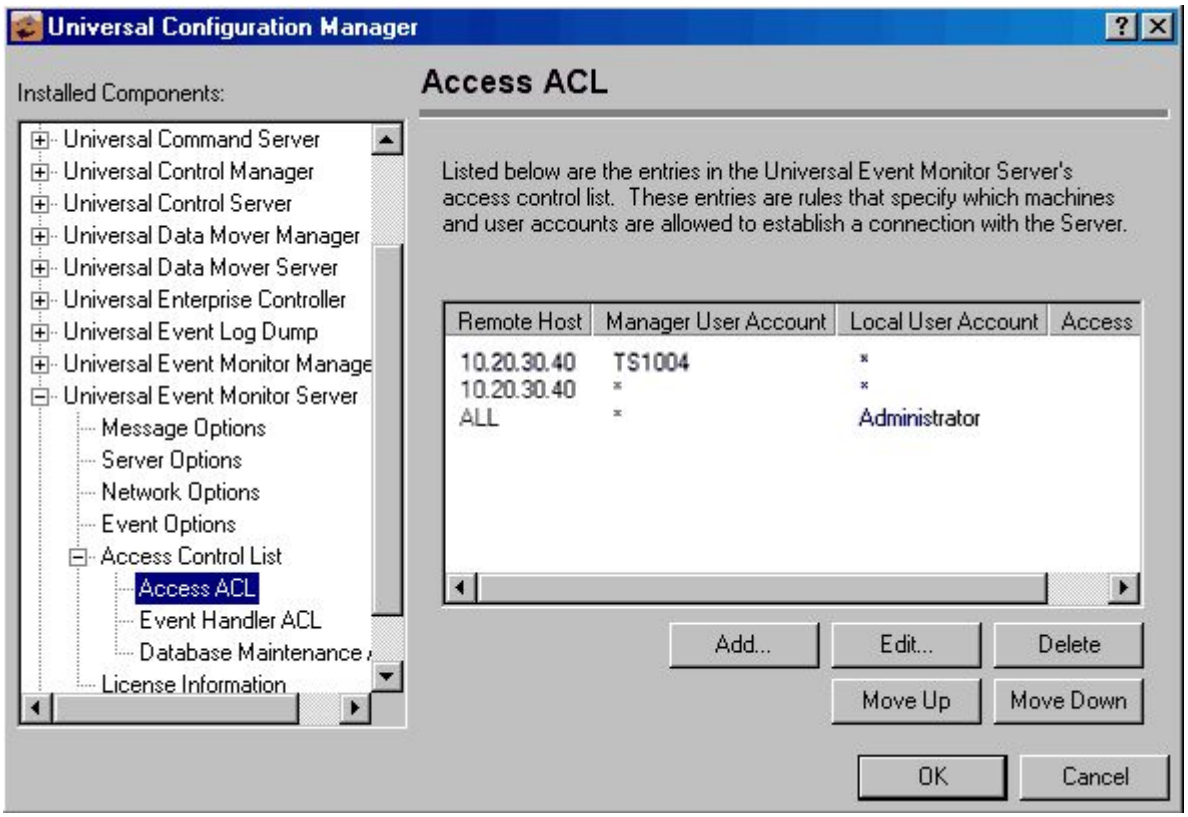
4.3.4.18 Universal Event Monitor Server for Windows - UACL Example

4.3.4.18.1 Universal Event Monitor Server for Windows

Although UACL files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to update UACL entries. From there, ACL entries can be added, changed, deleted, or sorted (rules are applied in the order in which they are listed).

The following figure illustrates an example. The set of ACL entries only allows connections from host 10.20.30.40 if the user on that host is TS1004. All other remote users will be blocked.

- TS1004 may run processes on the local system using any user account, provided the correct password is supplied.
- No processes may be run with Universal Event Monitor using the Administrator account on the local system, regardless of where the request originated.



4.3.4.18.1.1 Components

Universal Event Monitor

4.3.4.19 Universal Event Monitor Server for UNIX - UACL Example

4.3.4.19.1 Universal Event Monitor Server for UNIX

The following set of rules permit services for the subnet 10.20.30 and denies all other connections unless an X.509 certificate is presented that maps to certificate ID operations.

```
uem_access      10.20.30.,*,*,allow,auth
uem_access      ALL,*,*,deny,auth
```

The following set of rules effectively permit connections from any host but has limited access from host 10.20.30.40 to user TS1004 on that host.

- No host can monitor events as local user root.
- User TS1004 on host 10.20.30.40 can monitor events as local user tsup1004 without providing the password.
- Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

```
uem_access      10.20.30.40,TS1004,tsup1004,allow,noauth
uem_access      10.20.30.40,TS1004,*,allow,auth
uem_access      10.20.30.40,*,*,deny,auth
uem_access      ALL,*,root,deny,auth
```

4.3.4.19.1.1 Components

Universal Event Monitor Server for UNIX

4.4 X.509 Certificates

4.4.1 Overview

A certificate is an electronic object that identifies an entity. It is analogous to a passport in that it must be issued by a party that is trusted by all who accept the certificate. Certificates are issued by trusted parties called Certificate Authorities (CA's). For example, VeriSign Inc. is a CA that most parties trust. We all have faith that a trusted CA takes the necessary steps to confirm the identity of a user before issuing the user a certificate.

Certificate technology is based on public/private key technology. There are a few different types of public/private keys: RSA, DH, and DSS. As their name denotes, the private key must be kept private, like a password. The public key can be given to anyone or even published in a newspaper.

A property of public/private keys is that data encrypted with one can be decrypted only with the other. Therefore, if a person wants to send you a secret message, that person must encrypt the data with your public key, which everyone has. However, since you are the only one with your private key, you are the only one who can decrypt it. If you want to send someone a message, such as a request for $100,000 purchase, you can "sign" it with your private key. Signing does not encrypt the data. Once a person receives your request, that person can verify it is from you by verifying your electronic signature with your public key.

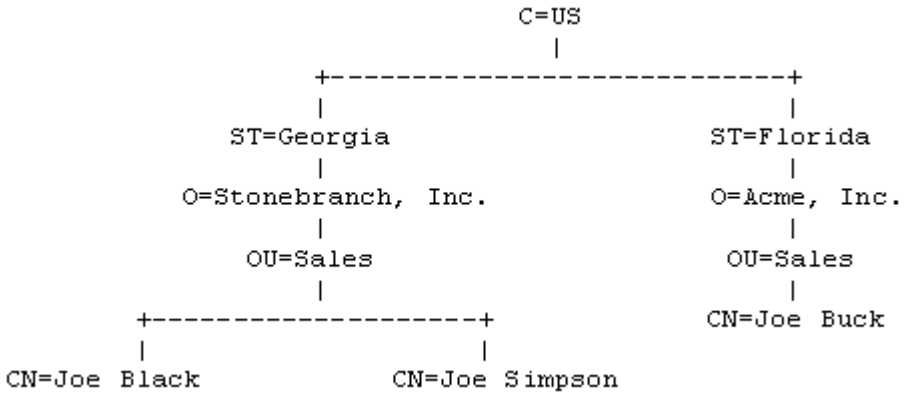
A certificate ties a statement of identity to a public key. Without the public key, the certificate is meaningless. Possession of a certificate alone does not prove your identity. You must have the corresponding private key. The two together prove your identity to any third party that trusts the CA that issued your certificate. This is a key point; if you do not trust the CA that signed a certificate, you cannot trust the certificate.

Since certificates originally were designed to be used for internet authentication, global directory technologies were developed to make them available via the internet. This directory technology is known as X.500 Directory Access Protocol. Later LDAP was introduced by Netscape to make it Lightweight Directory Access Protocol.

X.500 divides the world into a hierarchical directory. A person's identity is located by traversing down the hierarchy until it reaches the last node. Each node in the hierarchy consists of a type of object, such as a country, state, company, department, or name.

4.4.2 Sample Certificate Directory

The following figure provides a sample diagram of a small X.500 directory.



The keywords listed on each node are referred to as a Relative Distinguished Name (RDN). A person is identified by a Distinguished Name (DN). The DN value for Joe Black is `C=US/ST=Georgia/O=Stonebranch, Inc./OU=Sales/CN=Joe Black`.

A certificate is composed of many fields and possible extensions. Many of the most popular fields are specified as X.500 DN values.

4.4.3 Certificates Examples

Creating Certificates Examples provides examples that illustrate how to use Universal Certificate.

The examples provide the command line options only so that they can be used easily in any environment.

4.4.4 Additional Information

The following pages provide detailed information for X.509 Certificates:

- Sample X.509 Certificate
- SSL/TLS Peer Authentication

4.4.5 Sample X.509 Certificate

- Sample X.509 Certificate
- Certificate Fields

4.4.5.1 Sample X.509 Certificate

The following figure illustrates a sample X.509 version 3 certificate for Joe Buck at the Acme corporation.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:02:03:04:05:06:07:08
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, ST=Florida, O=Acme, Inc., OU=Security, CN=CA Authority/
emailAddress=ca@acme.com
        Validity
            Not Before: Aug 20 12:59:55 2013 GMT
            Not After : Aug 20 12:59:55 2013 GMT
        Subject: C=US, ST=Florida, O=Acme, Inc., OU=Sales, CN=Joe Buck
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public Key: (1024 bit)
                    Modulus (1024 bit):
                        00:be:5e:6e:f8:2c:c7:8c:07:7e:f0:ab:a5:12:db:
                        fc:5a:1e:27:ba:49:b0:2c:e1:cb:4b:05:f2:23:09:
                        77:13:76:57:08:29:45:29:d0:db:8c:06:4b:c3:10:
                        88:e1:ba:5e:6f:1e:c0:2e:42:82:2b:e4:fa:ba:bc:
                        45:e9:98:f8:e9:00:84:60:53:a6:11:2e:18:39:6e:
                        ad:76:3e:76:8d:1e:b1:b2:1e:07:97:7f:49:31:35:
                        25:55:0a:28:11:20:a6:7d:85:76:f7:9f:c4:66:90:
                        e6:2d:ce:76:45:66:be:56:aa:ee:93:ae:10:f9:ba:
                        24:fe:38:d0:f0:23:d7:a1:3b
                    Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Alternative Name:
                email:joe.buck@acme.com
    Signature Algorithm: md5WithRSAEncryption
        a0:94:ca:f4:d5:4f:2d:da:a8:6d:e3:41:6e:51:83:57:b3:b5:
        31:95:32:b6:ca:7e:d1:4f:fb:01:82:db:23:a0:39:d8:69:71:
        31:9c:0a:3b:ce:f6:c6:e2:5c:af:23:f0:d7:ee:87:3e:8a:7b:
        40:03:39:64:a1:8c:29:7d:5b:99:93:fa:23:19:e1:e4:ac:4d:
        13:0f:de:ad:51:27:e3:4e:4b:9f:40:4c:05:fd:f2:82:09:3e:
        46:05:f0:ad:cc:f7:78:25:3e:11:f8:ca:b6:df:f7:37:57:9b:
        63:00:d0:b5:b5:18:ec:38:76:d2:85:a3:c7:24:21:47:ee:f2:
        8c:0d
```

Note

The contents of a certificate file does not look like the information in this figure, which is produced by a certificate utility that uses the certificate file as input. Certificates can be saved in multiple file formats, so their file contents will look very different.

4.4.5.2 Certificate Fields

A certificate is composed of many fields.

The following table describes the main certificate fields.

Field or Section	Description
Version	X.509 certificates come in two versions: 1 and 3.
Serial Number	CA is required to provide each certificate it issues a unique serial number. The serial number is not unique for all certificates, only for the certificates issued by each CA.
Issuer	DN name of the CA that issued the certificate.
Validity	Starting and ending date for which this certificate is valid.
Subject	Identity of the certificate. A certificate may identify a person or a computer. In this case, the certificate identifies Joe Buck in the Sales organization of the Acme company in the state of Florida in the United States.
Public Key	Public key associated with the certificate identity.
X509v3 Extensions	X.509 version 3 introduced this section so that additional certificate fields may be added. In this case, the identity's email address is included as a Subject Alternative Name field. **Note** This section is not available in X.509 version 1.
Signature	CA's digital signature of the certificate.

4.4.6 SSL/TLS Peer Authentication

- Overview
- Certificate Verification
- Certificate Revocation
- Certificate Identification
- Certificate Support
- Sample Set-up for Universal Command Peer Authentication of Universal Broker
    - Certificate

4.4.6.1 Overview

The SSL/TLS protocol utilizes X.509 certificates to perform peer authentication. For example, a Universal Command Manager may want to authenticate that it is connected to the correct Universal Broker.

Peer authentication is performed by either one or both of the programs involved in the network session. If a Manager wants to authenticate the Broker to which it connects, the Broker will send its certificate to the Manager for the Manager to authenticate. If the Broker wants to authenticate the Manager, the Manager sends its certificate to the Broker.

Certificate authentication is performed in the following steps:

1. Check that the peer certificate is issued by a trusted CA.
2. Check that the certificate has not been revoked by the CA.
3. Check that the certificate identifies the intended peer.

If a step fails, the network session is terminated immediately.

4.4.6.2 Certificate Verification

The Universal Agent component must be configured with a list of trusted CA certificates. When a peer certificate is received, the trusted CA certificates are used to verify that the peer certificate is issued by one of the trusted CA's.

The trusted CA certificate list must be properly secured so that only authorized accounts have update access to the list. Should the trusted CA list become compromised, there is a possibility that an untrusted CA certificate was added to the list.

The CA certificate list configuration option is CA_CERTIFICATES. It specifies a PEM-formatted file that contains one or more CA certificates used for verification.

Should a peer certificate not be signed by a trusted CA, the session is immediately terminated.

4.4.6.3 Certificate Revocation

After a certificate is verified to have come from a trusted CA, the next step is to check if the CA has revoked the certificate. Since a certificate is held by the entity for which it identifies, a CA cannot take a certificate back after it is issued. So if a CA needs to revoke a certificate for some reason, it issues a list of revoked certificates referred to as the Certificate Revocation List (CRL). A program that validates certificates must have access to the latest CRL issued by the CA.

The CERTIFICATE_REVOCATION_LIST configuration option specifies the PEM-formatted file that contains the CRL. This option is available in all Universal Agent components that utilize certificates.

4.4.6.4 Certificate Identification

After a certificate is validated as being issued by a trusted CA, and has not been revoked by the CA, the next step is to check that it identifies the intended peer.

A Universal Agent Manager validates a Broker certificate by the Broker host name, IP address, or the certificate serial number. The VERIFY_HOST_NAME configuration option is used to specify the host name or IP address that is identified in the Broker certificate. Each certificate signed by a CA must have a unique serial number for that CA. The VERIFY_SERIAL_NUMBER option is used to specify the serial number in the Broker certificate.

If certificate identification fails, the session is immediately terminated.

Universal Brokers work differently than the Managers. A Broker maps a peer certificate to a certificate ID. The certificate map definitions are part of the Universal Access Control List (UACL) definitions. At that point, the certificate ID is used by UACL definitions to control access to Broker and Server services.

4.4.6.5 Certificate Support

Many certificate authority applications, also known as Public Key Infrastructure (PKI) applications, are available. Universal Agent should be able to utilize any certificate in a PEM-formatted file. PEM (Privacy Enhanced Mail) is a common text file format used for certificates, private keys, and CA lists.

Universal Agent support X.509 version 1 and version 3 certificates.

Although implementing a fully featured PKI infrastructure is beyond the scope of Universal Agent and this documentation, some assistance is provided using the OpenSSL toolkit.

Universal Agent on most of the supported platforms utilize the OpenSSL toolkit for its SSL/TLS and certificate implementation. OpenSSL is delivered on most UNIX distributions and Windows distributions and also is available on the OpenSSL website.

Universal Agent supports z/OS System SSL on the IBM z/OS operating system as well as OpenSSL. System SSL interfaces directly with the RACF security product for certificate access. All certificates, CA and user certificates, and private keys must be stored in the RACF database to use System SSL.

The Universal Agent suite includes an X.509 certificate utility, Universal Certificate, to create certificates for use in the Universal Agent suite.

4.4.6.6 Sample Set-up for Universal Command Peer Authentication of Universal Broker

Step 1	Create a Self-Signed CA Request: `ucert -create request -request_file ca_req.pem -private_key_file ca_pkey.pem -country US -state GA -locality Alpharetta -organization Stonebranch -common_name Stonebranch`
Step 2	Create a CA Certificate: `ucert -create cert -request_file ca_req.pem -private_key_file ca_pkey.pem -cert_file ca_cert.pem -ca yes -not_after_date +3650`
Step 3	Create a Server Certificate Request: `ucert -create request -request_file ubr1_req.pem -private_key_file ubr1_pkey.pem -country US -state GA -locality Alpharetta -organization Stonebranch -common_name "l64agent"`
Step 4	Create a Server Certificate: `ucert -create cert -ca_cert_file ca_cert.pem -request_file ubr1_req.pem -private_key_file ca_pkey.pem -cert_file ubr1_cert.pem -not_after_date +3650`
Step 5	The following files are generated in Steps 1 - 4: • CA PKEY = `ca_pkey.pem` • CA CERT = `ca_cert.pem` • Server PKEY = `ubr1_pkey.pem` • Server CERT = `ubr1_cert.pem`
Step 6	Add Server CERT and PKEY to the target `ubroker.conf`: • `certificate /home/test/ubr1_cert.pem` • `private_key /home/test/ubr1_pkey.pem`

Step 7	Copy `ca_cert.pem` to the source server.
Step 8	Run the following command from the source server to test: `/opt/universal/bin/ucmd -host l64agent -userid test -pwd xxx -cmd "pwd" -level info -verify_host_name yes -ca_certs /home/test/ca_cert.pem`
Step 9	Use Universal Certificate to print the certificate and verify the certificate serial number: `ucert -print cert -cert_file ubr1_cert.pem` See #Certificate, below.
Step 10	Run following command from the source server to test: `/opt/universal/bin/ucmd -host l64agent -userid test -pwd xxx -cmd "pwd" -level info -verify_host_name yes -ca_certs /home/test/ca_cert.pem -verify_serial_number 0x28c91a7fb2f26649`

4.4.6.6.1 Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            28:c9:1a:7f:b2:f2:66:49
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=GA, L=Alpharetta, O=Stonebranch, CN=Stonebranch
        Validity
            Not Before: Feb  8 21:08:12 2016 GMT
            Not After : Feb  6 02:08:12 2026 GMT
        Subject: C=US, ST=GA, L=Alpharetta, O=Stonebranch, CN=l64agent
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d9:30:22:5b:b4:62:5c:d9:26:4b:16:02:cc:22:
                    65:b8:ed:89:2d:6e:94:f8:b4:51:2c:1b:b7:5b:63:
                    76:ce:c5:05:a6:a9:52:47:f2:56:5e:58:cd:f8:c6:
                    a9:1d:54:a6:52:9f:5c:95:4f:27:db:bd:6f:ba:cc:
                    23:17:67:aa:3a:12:1b:21:97:32:ce:bf:22:c2:1c:
                    2d:4b:a5:c4:99:18:38:96:48:06:9b:2b:98:df:76:
                    e3:92:af:86:21:76:ed:77:86:63:af:a2:71:c4:0e:


                    a8:ac:1d:dc:26:65:b0:ed:b0:06:50:4b:da:e4:01:


                    7a:49:7e:9b:38:1d:c7:2d:57


                Exponent: 3 (0x3)
```

```
    X509v3 extensions:

        X509v3 Basic Constraints:

            CA:FALSE

        X509v3 Subject Key Identifier:

            CA:8D:DB:15:B8:A9:42:EC:51:A2:B7:C3:19:76:F7:15:35:1D:C8:9E

        X509v3 Authority Key Identifier:

            DirName:/C=US/ST=GA/L=Alpharetta/O=Stonebranch/CN=Stonebranch

            serial:79:19:7A:72:ED:D5:1F:7B


        X509v3 Key Usage:

            Digital Signature, Non Repudiation, Key Encipherment

Signature Algorithm: sha1WithRSAEncryption

    b0:b3:0d:8c:06:fe:4a:b0:e8:46:fd:8f:d8:64:d1:5e:11:b3:

    68:43:34:28:08:4b:e0:62:39:c1:6c:06:76:f3:e5:9d:8c:4e:

    15:57:56:d7:bf:92:f3:cf:6a:c8:36:54:28:2d:f9:9f:ad:67:

    44:1a:2e:32:ad:8b:8a:a0:86:64:8d:76:a0:60:46:65:f0:62:

    1f:02:db:c7:7c:99:db:ad:5b:80:3e:e9:b2:88:19:23:15:e6:

    7a:1d:53:e3:51:60:2d:99:0c:20:08:5a:ae:0f:c8:d3:20:a4:

    31:91:8b:a7:c2:c8:7a:ab:6c:2d:18:7a:1e:95:4b:c0:3e:5f:
    f9:cf
```

4.5 Creating Certificates Examples

4.5.1 Examples

- Creating a Certificate Authority (CA) Certificate
- Creating a RSA Certificate
- Creating an ECDSA Certificate

4.5.2 Creating a Certificate Authority (CA) Certificate

4.5.2.1 Creating a Certificate Authority Certificate

The first step in creating a certificate hierarchy is creating the root Certificate Authority (CA) certificate. The CA certificate is used to issue user certificates.

A certificate is created by creating a certificate request and then having the CA validate and sign the certificate. Since we are creating a root CA certificate, there is no CA to sign the certificate request, so instead a self-signed certificate is created and the CA flag is set.

The following Universal Certificate command creates:

- Certificate request, which it writes it to file **req.pem**
- Private key, which it writes it to file **cakey.pem**

```
ucert -create request -request_file req.pem -private_key_file cakey.pem -country US
-state Maryland -locality Baltimore
      -organization "Acme, Inc." -common_name "Acme CA"
```

It is imperative that the private key file **cakey.pem** is secured so that no one other than the CA has read access. If unauthorized access is gained to the CA's private key, all certificates issued by the CA no longer can be trusted.

The following Universal Certificate command creates the CA certificate and writes it to file **cacert.pem**.

```
ucert -create cert -request_file req.pem -cert_file cacert.pem -private_key_file
cakey.pem -ca yes
```

The CA certificate, **cacert.pem**, must be made available to any system that wants to consider the certificates issued by the CA as valid.

4.5.2.1.1 Components

Universal Certificate

4.5.3 Creating a RSA Certificate

4.5.3.1 Creating a RSA Certificate

There are two steps in creating a RSA certificate:

1. First step is performed by the party that wants the certificate.
2. Second step is performed by the Certificate Authority (CA) that creates the certificate.

4.5.3.1.1 Step 1

Step one is creating the certificate request. The certificate request will then be sent to the CA that verifies the request and creates the certificate from the request. The command that creates the certificate request also creates a private key. The private key must be secured so that only the entity identified by the certificate request has read access.

The following Universal Certificate command creates:

- Certificate request, which it writes it to file **req.pem**
- Private key, which it writes it to file **pkey.pem**

```
ucert -create request -request_file req.pem -private_key_file pkey.pem -country US -state
Maryland -locality Baltimore
      -organization "Acme, Inc." -common_name "Joe Buck"
```

4.5.3.1.2 Step 2

Step two is for the CA to create a certificate from the request and sign it with the CA's private key.

The following Universal Certificate command creates the certificate and writes it to file **cert.pem**.

```
ucert -create cert -request_file req.pem -cert_file cert.pem -private_key_file cakey.pem
-ca_cert_file cacert.pem
```

4.5.3.1.3 Components

Universal Certificate

4.5.4 Creating an ECDSA Certificate

4.5.4.1 Creating an ECDSA Certificate

There are two steps in creating an ECDSA certificate:

1. First step is performed by the party that wants the certificate.
2. Second step is performed by the Certificate Authority (CA) that creates the certificate.

4.5.4.1.1 Step 1

Step one is creating the certificate request. The certificate request will then be sent to the CA that verifies the request and creates the certificate from the request. The command that creates the certificate request also creates a private key. The private key must be secured so that only the entity identified by the certificate request has read access.

The following Universal Certificate command creates:

- Certificate request, which it writes it to file **req.pem**
- Private key, which it writes it to file **pkey.pem**

```
ucert -create request -request_file req.pem -private_key_file pkey.pem -private_key_type
EC -country US -state Maryland -locality Baltimore -organization "Acme, Inc."
-common_name "Joe Buck"
```

4.5.4.1.2 Step 2

Step two is for the CA to create a certificate from the request and sign it with the CA's private key.

The following Universal Certificate command creates the certificate and writes it to file **cert.pem**.

```
ucert -create cert -request_file req.pem -cert_file cert.pem -private_key_file cakey.pem
-ca_cert_file cacert.pem
```

4.5.4.1.3 Components

Universal Certificate