Universal Products Utilities

User Guide

Universal Products

Version 3.2.0

util-user-3205

Universal Products Utilities

User Guide

Universal Products 3.2.0

Document Name	Universal Products Utilities 3.2.0 User Guide				
Document ID	**util-user-3205**				
UnIversal Products Utilities	**z/OS**	**UNIX**	**Windows**	**OS/400**	**HP NonStop***
Universal Certificate	√	√	√		
Universal Control	√	√	√	√	√
Universal Copy		√	√	√	√
Universal Database Dump	√	√	√		
Universal Database Load	√	√	√		
Universal Display Log File				√	
Universal Encrypt	√	√	√	√	√
Universal Event Log Dump			√		
Universal Message Translator	√	√	√	√	√
Universal Query	√	√	√	√	√
Universal Return Code			√		
Universal Spool List	√	√	√	√	
Universal Spool Remove	√	√	√	√	
Universal Submit Job				√	
Universal WTO	√				
* Utilities for Universal Products 2.1.1 is used on the HP NonStop operating system.					

Stonebranch Documentation Policy

Stonebranch, Inc.
950 North Point Parkway, Suite 200
Alpharetta, GA 30005 USA
Tel: (678) 366-7887
Fax: (678) 366-7717

SAP Certified Integration

Summary of Changes

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-3205) October 29, 2010

- Modified the return code of the script to be equal to the return code of the **backup.exe** program in Section 14.3.1 Universal Command Manager for z/OS executing Universal Return Code within a Script.

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-3204) September 8, 2009

Universal Spool 3.2.0.3

- Added the following sections:
  - Section 15.2.7 zFS Support in Chapter 15 Universal Spool List
  - Section 16.3.7 zFS Support in Chapter 16 Universal Spool Remove

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-3203) July 29, 2009

Universal Products Utilities 3.2.0.1 for OS/400

- Modified document for upgrade from Universal Products Utilities 3.1.1 for OS/400 to Universal Products Utilities 3.2.0 for OS/400, including:
  - Changed the following OS/400 names throughout the document:
    - Universal Broker subsystem name from **UBROKER** to **UNVUBR320**.
    - Universal Broker user profile name from **UBROKER** to **UNVUBR320**.
    - Universal Products installation library name from **UNIVERSAL** to **UNVPRD320**.

    - Universal Products spool library name from **UNVSPOOL** to **UNVSPL320**.
    - Universal Products temporary directory from **UNVTMP** to **UNVTMP320**.
- Added a Universal Products for OS/400 Commands section, describing OS/400 command names and use of the Change Release Tag program (**UCHGRLS**) to change those names, in:
    - Section 4.5 Universal Control Manager for OS/400
    - Section 5.3 Universal Copy for OS/400
    - Chapter 8 Universal Display Log File
    - Section 9.4 Universal Encrypt for OS/400
    - Section 13.4 Universal Query for OS/400
    - Section 11.5 Universal Message Translator for OS/400

Universal Control 3.2.0.1 for OS/400

- Added the following configuration options in Section 4.5 Universal Control Manager for OS/400 of Chapter 4 Universal Control:
    - ACTIVITY_MONITORING
    - CERTIFICATE_REVOCATION_LIST
    - EVENT_GENERATION
    - PLF_DIRECTORY

Universal Query 3.2.0.1 for OS/400

- Added the following configuration options in Section 13.4 Universal Query for OS/400 of Chapter 13 Universal Query:
    - COMMAND_ID
    - COMPONENT_ID
    - MANAGERS
    - PLF_DIRECTORY

Universal Spool 3.2.0.1 for OS/400

- Added Section 15.4 Universal Spool List for OS/400 in Chapter 15 Universal Spool List.
- Added Section 16.5 Universal Spool Remove for OS/400 in Chapter 16 Universal Spool Remove.

Universal Submit Job 3.2.0.1 for OS/400

- Added the following SBMJOB Encapsulated configuration option in Chapter 17 Universal Submit Job:
    - INLASPGRP

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-3202) April 1, 2009

- Moved the Licenses and Copyrights appendix to the Universal Products 3.2.0 Installation Guide.

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-3201) September 5, 2008

- Added toll-free telephone number for North America in Appendix A Customer Support.

Changes for Universal Products Utilities 3.2.0 User Guide (util-user-320) May 16, 2008

Universal Products 3.2.0.0

- Added support for the following features:
  - Universal Query returns a variety of information on Universal Broker and the components managed by the Broker. Universal Query now can return information on Manager components running local to a Broker.
- Added the following chapters:
  - Chapter 2 Features, including Section 2.4 Universal Configuration Manager
  - Chapter 6 Universal Database Dump
  - Chapter 7 Universal Database Load
  - Chapter 12 Universal Products Install Merge
- Modified Chapter 3 Universal Certificate
- Added the following configuration options in Chapter 3 Universal Certificate:
  - CERT_DB
  - CRL_FILE
  - CRL_FORMAT
  - NEXT_UPDATE_DAYS
  - NEXT_UPDATE_HOURS
  - REVOKE
  - REVOKE_REASON
  - STATE
  - TRANSPORT_FILE
  - TRANSPORT_FILE_PWD
  - VERIFY
- Added the following configuration options in Chapter 10 Universal Event Log Dump:
  - INSTALLATION_DIRECTORY
  - LOG_DIRECTORY

    - NLS_DIRECTORY
- Added the following configuration options in Chapter 13 Universal Query:
    - BIF_DIRECTORY
    - COMMAND_ID
    - COMPONENT _ID
    - MANAGERS
    - PLF_DIRECTORY
    - NLS_DIRECTORY
    - SYSTEM_ID

Contents

List of Figures

List of Tables

Preface

Document Structure

This document is written using specific conventions for text formatting and according to a specific document structure in order to make it as useful as possible for the largest audience. The following sections describe the document formatting conventions and organization.

Format

Starting with the Universal Products 3.2.0 release, a Universal Products Utilities User Guide has been created. Formerly, information on Universal Products Utilities was documented in the Universal Command User Guide.

Additionally, links to detailed information in a companion document, the Universal Products Utilities Reference Guide, have been created in the user guide.

In order for the links between these documents to work correctly:

- Place the documents in the same folder.
- In Adobe Reader / Adobe Acrobat, de-select **Open cross-document link in same window** in the **General** category of your **Preferences** dialog (selected from the **Edit** menu).

Conventions

Specific text formatting conventions are used within this document to represent different information. The following conventions are used.

Typeface and Fonts

`This Font` identifies specific names of different types of information, such as file names or directories (for example, **`\abc\123\help.txt`**).

Command Line Syntax Diagrams

Command line syntax diagrams use the following conventions:

Convention	Description
`bold monospace font`	Specifies values to be typed verbatim, such as file / data set names.
`italic monospace font`	Specifies values to be supplied by the user.
[ ]	Encloses configuration options or values that are optional.
{ }	Encloses configuration options or values of which one must be chosen.
|	Separates a list of possible choices.
. . .	Specifies that the previous item may be repeated one or more times.
BOLD UPPER CASE	Specifies a group of options or values that are defined elsewhere.

Table P.1 Command Syntax

Operating System-Specific Text

Most of this document describes the product in the context of all supported operating systems. At times, it is necessary to refer to operating system-specific information. This information is introduced with a special header, which is followed by the operating system-specific text in a different font size from the normal text.

z/OS

This text pertains specifically to the z/OS line of operating systems.

This text resumes the information pertaining to all operating systems.

Tips from the Stoneman

Stoneman's Tip

Look to the Stoneman for suggestions
or for any other information
that requires special attention.

Vendor References

References are made throughout this document to a variety of vendor operating systems. We attempt to use the most current product names when referencing vendor software.

The following names are used within this document:

- **z/OS** is synonymous with IBM z/OS and IBM OS/390 line of operating systems.
- **Windows** is synonymous with Microsoft's Windows 2000 / 2003 / 2008, Windows XP, Windows Vista, and Windows 7 lines of operating systems. Any differences between the different systems will be noted.
- **UNIX** is synonymous with operating systems based on AT&T and BSD origins and the Linux operating system.
- **OS/400** is synonymous with IBM OS/400, IBM i/5, and IBM i operating systems.
- **AS/400** is synonymous for IBM AS/400, IBM iSeries, and IBM System i systems.

Note: These names do not imply software support in any manner. For a detailed list of supported operating systems, see the Universal Products 3.2.0 Installation Guide.

Document Organization

The document is organized into the following chapters:

- Overview (Chapter 1)
  General architectural and functional overview of the Universal Products Utilities.
- Features (Chapter 2)
  Description of Universal Products Utilities features, including configuration methods and network protocols.
- Universal Certificate (Chapter 3)
  Description of the Universal Certificate utility for all relevant operating systems.
- Universal Control (Chapter 4)
  Description of the Universal Control utility for all relevant operating systems.
- Universal Copy (Chapter 5)
  Description of the Universal Copy utility for all relevant operating systems.
- Universal Database Dump (Chapter 6)
  Description of the Universal Database Dump utility for the z/OS operating system.
- Universal Database Load (Chapter 7)
  Description of the Universal Database Load utility for all relevant operating systems.
- Universal Display Log File (Chapter 8)
  Description of the Universal Display Log file utility for the OS/400 operating system.
- Universal Encrypt (Chapter 9)
  Description of the Universal Encrypt utility for all relevant operating systems.
- Universal Event Log Dump (Chapter 10)
  Description of the Universal Event Log Dump utility for the Windows operating system.
- Universal Message Translator (Chapter 11)
  Description of the Universal Message Translator utility for all relevant operating systems.
- Universal Products Install Merge (Chapter 12)
  Description of the Universal Products Install Merge utility for all relevant operating systems.
- Universal Query (Chapter 13)
  Description of the Universal Query utility for all relevant operating systems.
- Universal Return Code (Chapter 14)
  Description of the Universal Return Code utility for the Windows operating system.
- Universal Spool List (Chapter 15)
  Description of the Universal Spool List utility for all relevant operating systems.
- Universal Spool Remove (Chapter 16)
  Description of the Universal Spool Remove utility for all relevant operating systems.
- Universal Submit Job (Chapter 17)
  Description of the Universal Submit Job utility for the OS/400 operating system.
- Universal Write-to-Operator (Chapter 18)
  Description of the Universal Write-to-Operator utility for the z/OS operating system.
- Customer Support (Appendix A)
  Customer support contact information for Universal Products Utilities.

Chapter 1
Overview

1.1 Introduction

This chapter provides general information on Universal Products Utilities.

Universal Products utilities are installed as part of each Universal Products package.

Some utilities are operating-system specific; they cannot be used on any operating system. The individual chapters in this document provide a complete description of each utility and identify the operaitng systems on which they can be used.

1.2 Types of Universal Products Utilities

Table 1.1, below, provides a description of each Universal Products utility.

Each Utility Name in the table is a link to its chapter in this document.

Utility Name	Description
Universal Certificate	Creates digital certificates and private keys, which Universal Products programs can use to securely identify users and computer systems.
Universal Control	Provides the ability to start and stop Universal Products components, and to refresh Universal Products configuration data.
Universal Copy	Provides a means to copy files from either manager-to-server or server-to-manager.
Universal Database Dump	Berkeley **db_dump** utility tailored specifically for Universal Products databases.
Universal Database Load	Berkeley **db_load** utility tailored specifically for Universal Products databases.
Universal Display Log File	Reads job log files; formats and writes job logs to standard out.
Universal Encrypt	Encrypts the contents of command files into an unintelligible format (for privacy reasons).
Universal Event Log Dump	Selects records from one of the Windows event logs and writes them to a specified output file.
Universal Message Translator	Translates error messages into return (exit) codes based on a user-defined translation table.
Universal Products Install Merge	Merges options and values from one Universal Products configuration or component definition file into another.
Universal Query	Queries any Universal Broker for Broker-related and active component-related information.
Universal Return Code	Performs the function of ending a process with a return code that is equal to its command line argument.
Universal Spool List	Provides the ability to list Universal Spool database records.
Universal Spool Remove	Provides the ability to remove component records from the Universal Command and Universal Event Monitor Spool databases.
Universal Submit Job	Encapsulates the IBM Submit Job (SBMJOB) command.
Universal Write-to-Operator	Issues Write-toOperator and Write-to-Opertator-with_Reply messages.

Table 1.1 Universal Products Utilities

Chapter 2
Features

2.1 Overview

This chapter provides information on Universal Products Utilities features that apply to all operating systems.

- Configuration
- Remote Configuration
- Universal Configuration Manager
- Network Data Transmission
- Universal Access Control List
- Message and Audit Facilities
- X.509 Certificates

2.2 Configuration

Product configuration consists of specifying options that control product behavior and resource allocation.

- An example of configurable product behavior is whether or not data transferred over the network is compressed.
- An example of configurable resource allocation is the directory location in which the product creates its log files.

Each option is comprised of a pre-defined parameter, which identifies the option, and one or more values. The format of the parameter depends on the method being used to specify the option.

Although there are many configurable product options, Universal Products, in general, are designed to require minimal configuration and administration. The default options will work very well in most environments. When local requirements do require a change in product configuration, there are multiple methods available to configure the products in order to meet your needs.

2.2.1 Configuration Methods

All Stonebranch Inc. Universal Products provide a consistent and flexible method of configuration. An operating system's native configuration methods, such as configuration files, are utilized in order to integrate with existing system management policies and procedures for the platform.

Depending on specific Universal Products, and the operating system on which it is being run, product configuration is performed by one or more methods. These configuration methods, in their order of precedence, are:

1. Command Line
2. Command Line File
3. Environment Variables
4. Configuration File

This order of precedence means that a command option specified on the command line overrides the same option specified in a command file, which overrides the same option specified with an environment variable, which overrides the same option specified with a configuration file keyword.

Note: For security reasons, not all options can be overridden.

2.2.2 Command Line

Command line options affect one instance of a program execution. Each time that you execute a program, command line options let you tailor the behavior of the program to meet the specific needs for that execution.

Command line options are the highest in order of precedence of all the configuration methods (see Section 2.2.1 Configuration Methods). They override the options specified using all other configuration methods, except where indicated.

Command line options consist of:

- Parameter (name of the option)
- Value (pre-defined or user-defined value of the option)

The command line syntax depends, in part, on the operating system, as noted below.

An value may or may not be case-sensitive, depending on what it is specifying. For example, if a value is either **yes** or **no**, it is not case-sensitive. It could be specified as **YES**, **Yes**, or **yes**. However, if a value specifies a directory name or file name, it would be case-sensitive if the operating system's file system is case-sensitive.

If an option is specified more than once on the command line, the last instance of the option specified is used.

z/OS

z/OS command line options are specified in the JCL EXEC statement PARM keyword or on the SYSIN ddname. The PARM keyword is used to pass command line options to the program being executed with the EXEC statement.

Command line options are prefixed with a dash (**-**) character. For many options, there are two different forms in which they can be specified:

- Short form: one case-sensitive character
- Long form: two or more case-insensitive characters

The parameter and value must be separated by at least one space.

Example command line options specified in the PARM value follow:

Short form:

```
PARM='-l INFO -G yes'
```

Long form:

```
PARM='-LEVEL INFO -LOGIN YES'
```

As noted above, z/OS command line options also can be specified on the SYSIN ddname. This is the easiest and least restrictive place to specify options, since the PARM values are limited in length. The options specified in the SYSIN ddname have the same syntax. Options can be specified on one line or multiple lines. The data set or inline data allocated to the SYSIN ddname cannot have line numbers in the last 8 columns (that is, all columns of the records are used as input).

UNIX, Windows, and HP NonStop

UNIX, Windows, and HP NonStop command line options are prefixed with a dash (**-**) character, and alternatively on Windows, the slash (**/**) character.

For many options, there are two different forms in which they can be specified:

- Short form: one case-sensitive character.
- Long form: two or more case insensitive characters.

The parameter and value must be separated by at least one space or tab character.

Example command line options follow:

Short form:

```
-l info -G yes
```

Long form:

```
-level info -login yes
-LEVEL info -LoGiN YES
```

OS/400

OS/400 command line options use the native conventions for Command Language (CL) commands. The option name is specified as a CL parameter with its value enclosed in parentheses.

Example command line options follow:

Command line options:

```
MSGLEVEL(INFO) COMPRESS(*YES)
```

All of the Stonebranch Inc. Universal Products provide OS/400-style command panels. The panels are accessed by entering the command name on the command line and pressing the F4 (PROMPT) key.

2.2.3 Command Line File

The command line file contains command line options specified in a file. The command line file enables you to save common command line options in permanent storage and reference them as needed.

The command line file is the second to highest in the precedence order after command line options (see Section 2.2.1 Configuration Methods).

Individual command line options can be specified on one or multiple lines. Blank lines are ignored. Lines starting with the hash (**#**) character are ignored and can be used for comments.

The command line file can be encrypted if it is necessary to secure the contents.

Note: If the contents of the file contain sensitive material, the operating system's native file and user security facilities should be used in addition to the file encryption provided by the Universal Products.

In order to use a command line file, either of the following is used:

- COMMAND_FILE_PLAIN option is used to specify the command line file name.
- COMMAND_FILE_ENCRYPTED option is used to specify the encrypted command line file name.

2.2.4 Environment Variables

Environment variables, like command line options, allow options to be specified for one instance of a program execution. Each time that you execute a program, environment variables allow you to tailor the behavior of the program to meet the specific needs for that execution.

Environment variables are the third highest in the precedence order, after command line file options (see Section 2.2.1 Configuration Methods).

Each operating system has its own unique method of setting environment variables.

All environment variables used by Universal Products are upper case and are prefixed with a product identifier consisting of three or four characters. The product sections specify the value of the environment variables. Values are case-sensitive.

z/OS

Environment variables are specified in the JCL EXEC statement PARM keyword. Environment variables are part of the IBM Language Environment (LE) and as such are specified as LE runtime options. The PARM value is divided into LE options and application options by a slash (/) character. Options to the left of the slash are LE options and options to the right are application options.

Example of setting an environment variable:

Set option UCTLEVEL to a value of INFO:

```
PARM='ENVAR("UCTLEVEL=INFO")/'
```

UNIX

Environment variables in UNIX are defined as part of the shell environment. As such, shell commands are used to set environment variables. The environment variable must be exported to be used be a called program.

Example of setting an environment variable:

Set option UCTLEVEL to a value of INFO in a bourne, bash, or korn shell:

```
UCTLEVEL=INFO
export UCTLEVEL
```

Windows

Environment variables in Windows are defined as part of the Windows console command environment. As such, console commands are used to set environment variables.

Example of setting an environment variable:

Set option UCTLEVEL to a value of INFO:

```
SET UCTLEVEL=INFO
```

OS/400

Environment variables in OS/400 are defined with Command Language (CL) commands for the current job environment.

Example of setting an environment variable:

Set option UCTLEVEL to a value of INFO:

```
ADDENVVAR ENVVAR(UCTLEVEL) VALUE(INFO)
```

HP NonStop

Environment variables in HP NonStop are defined with HP NonStop Advanced Command Language (TACL) commands for the current job environment.

Example of setting an environment variable:

Set option UCTLEVEL to a value of INFO:

```
PARAM UCTLEVEL INFO
```

2.2.5 Configuration File

Configuration files are used to specify system-wide configuration values. They are last in precedence order for specifying configuration options (see Section 2.2.1 Configuration Methods).

For most Universal Products, some options can be specified only in a configuration file, while other options can be overridden by individual command executions. The Stonebranch, Inc. documentation for each product identifies these options.

If an option is specified more than once in a configuration file, the last option specified is used.

All configuration files on a system are maintained by the local Universal Broker. The Universal Broker serves the configuration data to other Universal Products running on the local system. The one exception is Universal Enterprise Controller (UEC). UEC directly reads its own configuration files.

The Universal Broker reads the configuration files when it first starts or when it receives a REFRESH command from Universal Control or Universal Enterprise Controller. Any changes made to a configuration file are not in effect until the Broker is recycled or receives a REFRESH command.

Universal Product components do not read the configuration files themselves. When a component starts, it first registers with the locally running Universal Broker. As part of the registration process, the Broker returns the components configuration data.

When the Universal Broker is operating in managed mode, the configuration information for the various Universal Products is "locked down" and can be modified or viewed only via the Universal Management Console (see Section 2.3.2 Managed Mode).

z/OS

Configuration files are members of a PDSE. The data set record format is fixed or fixed block with a record length of 80. No line numbers can exist in columns 72-80. All 80 columns are processed as data.

All configuration files are installed in the **UNVCONF** library.

See Section 2.2.6 Configuration File Syntax for the configuration file syntax.

UNIX

Configuration files are regular text files on UNIX.

Universal Broker searches for the configuration files in a fixed list of directories. The Broker will use the first configuration file that it finds in its search. The directories are listed below in the order they are searched:

Directory	Notes
/etc/opt/universal	
/etc/universal	Installation default
/etc/stonebranch	Obsolete as of version 2.2.0
/etc	
/usr/etc/universal	
/usr/etc/stonebranch	Obsolete as of version 2.2.0
/usr/etc	

Table 2.1 UNIX Configuration File Directory Search

See Section 2.2.6 Configuration File Syntax for the configuration file syntax.

Windows

Although configuration 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 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 (see Section 2.4 Universal Configuration Manager).

OS/400

The configuration files on OS/400 are stored in a source physical file named UNVCONF in the UNVPRD320 library. The files can be edited with a text editor.

See Section 2.2.6 Configuration File Syntax for the configuration file syntax.

HP NonStop

The configuration files on HP NonStop are stored as EDIT files, file code 101, within the **$SYSTEM.UNVCONF** subvolume. The files can be edited with the EDIT editor.

See Section 2.2.6 Configuration File Syntax for the configuration file syntax.

2.2.6 Configuration File Syntax

Configuration files are text files that can be edited with any available text editor.

The following rules apply for configuration file syntax:

- Options are specified in a keyword / value format.
- Keywords can start in any column.
- Keywords must be separated from values by at least one space or tab character.
- Keywords are not case sensitive.
- Keywords cannot contain spaces or tabs.
- Values can contain spaces and tabs, but if they do, they must be enclosed in single (**'**) or double (**"**) quotation marks. Repeat the enclosing characters to include them as part of the value.
- Values case sensitivity depends on the value being specified. For example:
  - Directory and file names are case sensitive.
  - Pre-defined values (such as **yes** and **no**) are not case sensitive.
- Each keyword / value pair must be on one line.
- Characters after the value are ignored.
- Newline characters are not permitted in a value.
- Values can be continued from one line to the next either by ending the line with a:
  - Plus (**+**) character, to remove all intervening spaces.
  - Minus (**-**) character, to preserve all intervening spaces between the end of the line being continued and the beginning of the continuing line.

  Ensure that the line continuation character is the last character on a line.
- Comment lines start with a hash (**#**) character.
- Blank lines are ignored.

Note: If an option is specified more than once in a configuration file, the last option specified is used.

2.3 Remote Configuration

Universal Products can be configured remotely by Universal Enterprise Controller using the Universal Management Console (UMC) client application. UMC instructs the Universal Broker of a remote Universal Agent to modify the configurations of the Universal Products components managed by that Broker.

Universal Broker supports remote configuration in either of two modes:

1. Unmanaged Mode
2. Managed Mode

2.3.1 Unmanaged Mode

Unmanaged mode is the default mode of operations for Universal Broker. It allows a Universal Broker – and the Universal Products managed by that Universal Broker – to be configured either:

- Locally, by editing configuration files.
- Remotely, via UMC.

The system administrator for the machine on which a Universal Agent resides can use any text editor to modify the configuration files of the various local Universal Products.

Via UMC, selected users can modify all configurations of any Universal Agent, including the local Universal Agent. UMC sends the modified data to the Universal Broker of that agent, which Universal Broker then uses to update the appropriate configuration files.

If UMC sends modifications for a Universal Broker configuration, Universal Broker validates the modified data before it accepts it. If the data fails validation, Universal Broker does not update its configuration file.

If UMC sends modification to the configuration of any other Universal Products component, the Universal Broker updates the appropriate configuration file. The component will use this new configuration at its next invocation.

Note: If errors or invalid configuration values are updated via UMC for a component other than Universal Broker, the component may not run successfully until the configuration has be corrected.

2.3.2 Managed Mode

When a Universal Broker is operating in managed mode, the configuration information for all Universal Products components managed by that Universal Broker is "locked down." Universal Broker stores the information in a database file located within its specified spool directory. The information can be modified only via Universal Management Console (UMC).

From this point on, Universal Broker uses the database file – not the configuration files – to access configuration information. Any configuration changes made to the components – via UMC – are placed in the database file. Therefore, as long as Universal Broker stays in managed mode, the configuration files may no longer contain current or valid configuration information.

If managed mode is de-selected for the Universal Broker, it reads the database file where it stored the configuration information. Universal Broker uses this information to create and/or update configuration files for the components.

- If a configuration file exists in the configuration directory, it is overwritten.
- If a configuration file does not exist, it is created.

Note: Because of remote configuration and the desire to be able to "lock down" all product configurations, Universal Broker – and all Universal Products servers – no longer support the command line and environmental variables methods of specifying configuration options.

Selecting Managed Mode

The managed mode of operations for Universal Broker is selected via the Universal Enterprise Controller Administration client application.

(See the Universal Enterprise Controller 3.2.0 Client Applications guide for specific information on how to select managed mode.)

Figure 2.1, below, illustrates remote configuration for one Universal Agent in managed mode and one Universal Agent in unmanaged mode.

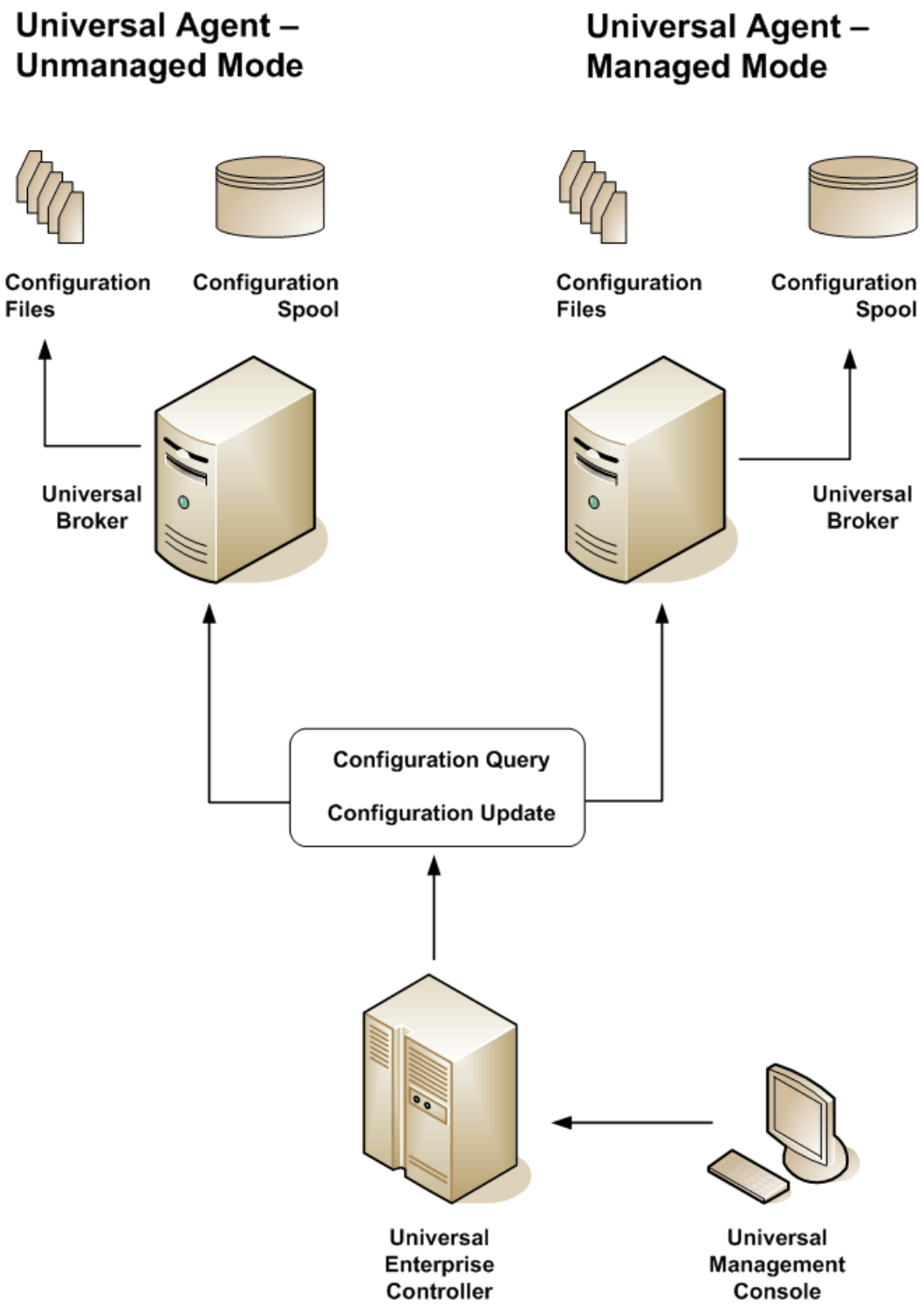

Figure 2.1 Remote Configuration - Unmanaged and Managed Modes of Operation

2.3.3 Universal Broker Startup

At Universal Broker start-up, in both managed and unmanaged modes, the Universal Broker configuration file is always read.

Unmanaged Mode

At Universal Broker start-up in unmanaged mode, Universal Broker reads the configuration files of all Universal Products components into its memory. The Universal Broker configuration file is used to define the Universal Broker configuration, just as all configuration files are used in unmanaged mode. Universal Broker updates its memory from the configuration files whenever Universal Control issues a REFRESH request.

Managed Mode

At Universal Broker start-up in managed mode, the Universal Broker configuration file points Universal Broker to the location of the configuration spool file, from which the Broker retrieves configuration information for all Universal Products. Universal Broker updates its memory from the configuration spool file and, automatically, after changes are made via UMC.

If more configuration information than needed is included in the Universal Broker configuration file at Universal Broker start-up, Universal Broker will update its running configuration with the information that it retrieved from the spool file. The configuration file that was used at start-up is made obsolete.

2.4 Universal Configuration Manager

The Universal Configuration Manager is a Universal Products graphical user interface application that enables you to configure all of the Universal Products that have been installed on a Windows operating system.

It is the recommended method of specifying configuration data that will not change with each command invocation. Universal Configuration Manager helps protect the integrity of the configuration file by validating all changes to configuration option values.

The configuration data for a Universal Products for Windows system is stored in the Windows configuration file.

2.4.1 Availability

Universal Configuration Manager is installed automatically on the Windows operating system as part of every Universal Products for Windows installation.

It is available to all user accounts in the Windows Administrator group.

Windows Vista

When opening the Universal Configuration Manager for the first time on Windows Vista, two new operating system features, the Program Compatibility Assistant (PCA) and User Account Control (UAC), may affect its behavior.

With these two features enabled, the expected Universal Configuration Manager behavior is as follows:

1. Universal Configuration Manager may issue the following error:

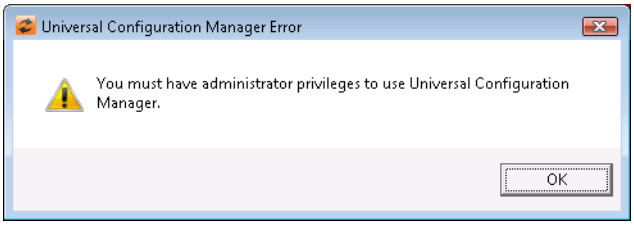

Figure 2.2 Universal Configuration Manager Error dialog - Windows Vista

2. Click **OK** to dismiss the error message.

   The Windows Vista Program Compatibility Assistant (PCA) displays the following dialog:

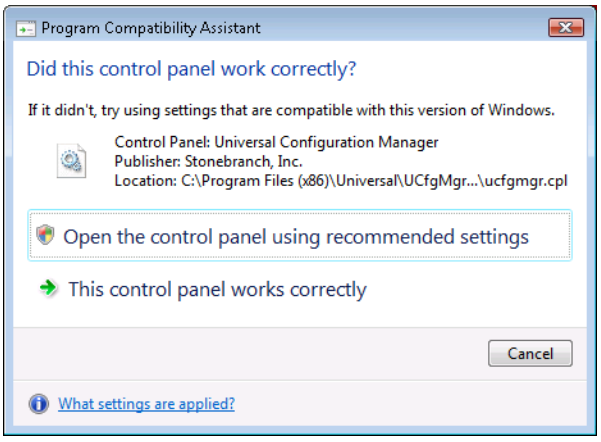

Figure 2.3 Windows Vista - Program Compatibility Assistant

3. To continue, select **`Open the control panel using recommended settings`**. This instructs the PCA to "shim" (Microsoft term) the Configuration Manager, establishing it as an application that requires elevated privileges.

   Windows Vista User Account Control (UAC) then displays a prompt seeking permission to elevate the logged-in account's access token.

4. Select **`Continue`** to give the account full administrative privileges.

   Subsequent attempts to open Universal Configuration Manager should result only in the UAC prompt.

2.4.2 Accessing the Universal Configuration Manager

To access the Universal Configuration Manager:

1. Click the **Start** icon at the lower left corner of your Windows operating system screen to display the Start menu.
2. Click (Settings/) Control Panel on the Start menu to display the Control Panel screen.
3. Select the Universal Configuration Manager icon to display the Universal Configuration Manager screen (see Figure 2.4).

Windows XP, Windows Vista, Windows Server 2008

Newer versions of Windows support a Control Panel view that places applet icons within categories. This "category view" may affect the location of the Universal Configuration Manager icon.

For example, the Windows XP Category View places the Universal Configuration Manager icon under the **Other Control Panel Options** link. Windows Vista and Windows Server 2008 place the icon within the **Additional Options** category.

If you have trouble locating the Universal Configuration Manager icon, simply switch to the Classic View to display all Control Panel icons at the same time.

64-bit Windows Editions

The Windows Control Panel places icons for all 32-bit applets under the **View x86 Control Panel Icons** (or, on newer versions, the **View 32-bit Control Panel Icons**) category, even when the Classic View is enabled.

When using the Category View, look for the 32-bit Control Panel applet icons in the **Additional Options** category.

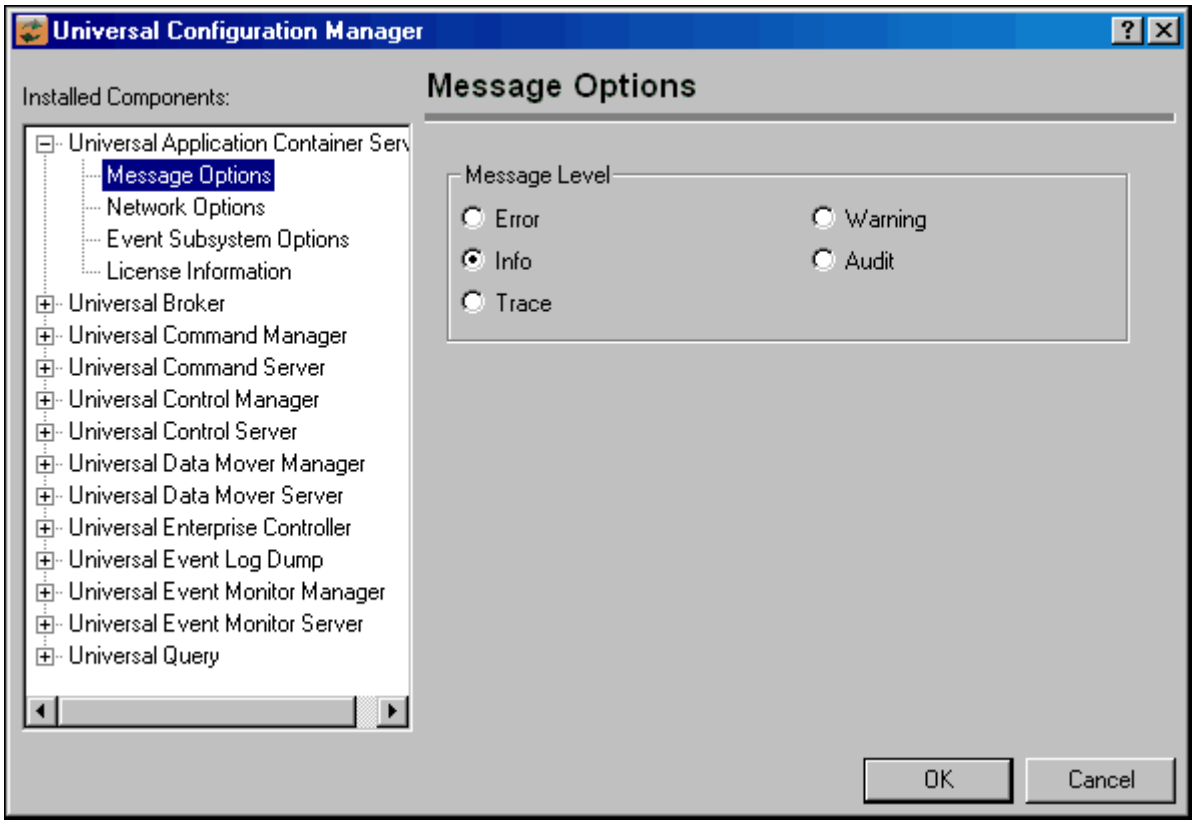

Figure 2.4 Universal Configuration Manager

Each Universal Configuration Manager screen contains two sections:

1. Left side of the screen displays the Installed Components tree, which lists:
   - Universal Products components currently installed on your system.
   - Property pages available for each component (as selected), which include one or more of the following:
     - Configuration options
     - Access control lists
     - Licensing information
     - Other component-specific information
2. Right side of the screen displays information for the selected component / page.

(By default, Universal Configuration Manager displays the first property page of the first component in the Installed Components tree.)

2.4.3 Navigating through Universal Configuration Manager

To display general information about a component, click the component name in the Installed Components list.

To display the list of property pages for a component, click the **+** icon next to the component name in the Installed Components list.

To display a property page, click the name of that page in the Installed Components list.

If a property page has one or more of its own pages, a **+** icon displays next to the name of that property page in the Installed Components list. Click that **+** icon to display a list of those pages.

In Figure 2.4, for example:

- List of property pages is displayed for Universal Broker.
- Message Options property page has been selected, and information for that property is displayed on the right side of the page.
- No **+** icon next to any of the property pages indicates that they do not have one or more of their own property pages.

2.4.4 Modifying / Entering Data

On the property pages, modify / enter data by clicking radio buttons, selecting from drop-down lists, and/or typing in data entry fields.

Some property pages provide panels that you must click in order to:

- Modify or adjust the displayed information.
- Display additional, modifiable information.

Note: You do not have to click the **OK** button after every modification or entry, or on every property page on which you have modified and/or entered data. Clicking **OK** just once, on any page, will save the modifications and entries made on all pages – and will exit Universal Configuration Manager (see Section 2.4.5 Saving Data.)

Rules for Modifying / Entering Data

The following rules apply for the modification and entry of data:

- Quotation marks are not required for configuration values that contain spaces.
- Edit controls (used to input free-form text values) handle conversion of any case sensitive configuration values. Except where specifically noted, values entered in all other edit controls are case insensitive.

2.4.5 Saving Data

To save all of the modifications / entries made on all of the property pages, click the **OK** button at the bottom of any property page. The information is saved to the Windows configuration file, and Universal Broker is automaticaly refreshed.

Clicking the **OK** button also exits the Universal Configuration Manager. (If you click **OK** after every modification, you will have to re-access Universal Configuration Manager.)

To exit Universal Configuration Manager without saving any of the modifications / entries made on all property pages, click the **Cancel** button.

2.4.6 Accessing Help Information

Universal Configuration Manager provides context-sensitive help information for the fields and panels on every Universal Products component options screen.

To access Help:

1. Click the question mark (**?**) icon at the top right of the screen.
2. Move the cursor (now accompanied by the **?**) to the field or panel for which you want help.
3. Click the field or panel to display Help text.
4. To remove the displayed Help text, click anywhere on the screen.

Windows Vista, Windows Server 2008

The Universal Configuration Manager's context-sensitive help is a WinHelp file, which Windows Vista and Windows Server 2008 does not support.

Microsoft offers the 32-bit WinHelp engine as a separate download from its website. If you require access to the Universal Configuration Manager's context-sensitive help, simply download and install the WinHelp engine.

2.4.7 Universal Products Utilities Installed Components

Universal Control Manager

Figure 2.5 illustrates the Universal Configuration Manager screen for the Universal Control Manager.

The Installed Components list identifies all of the Universal Control Manager property pages.

The text describes the selected component, Universal Control Manager.

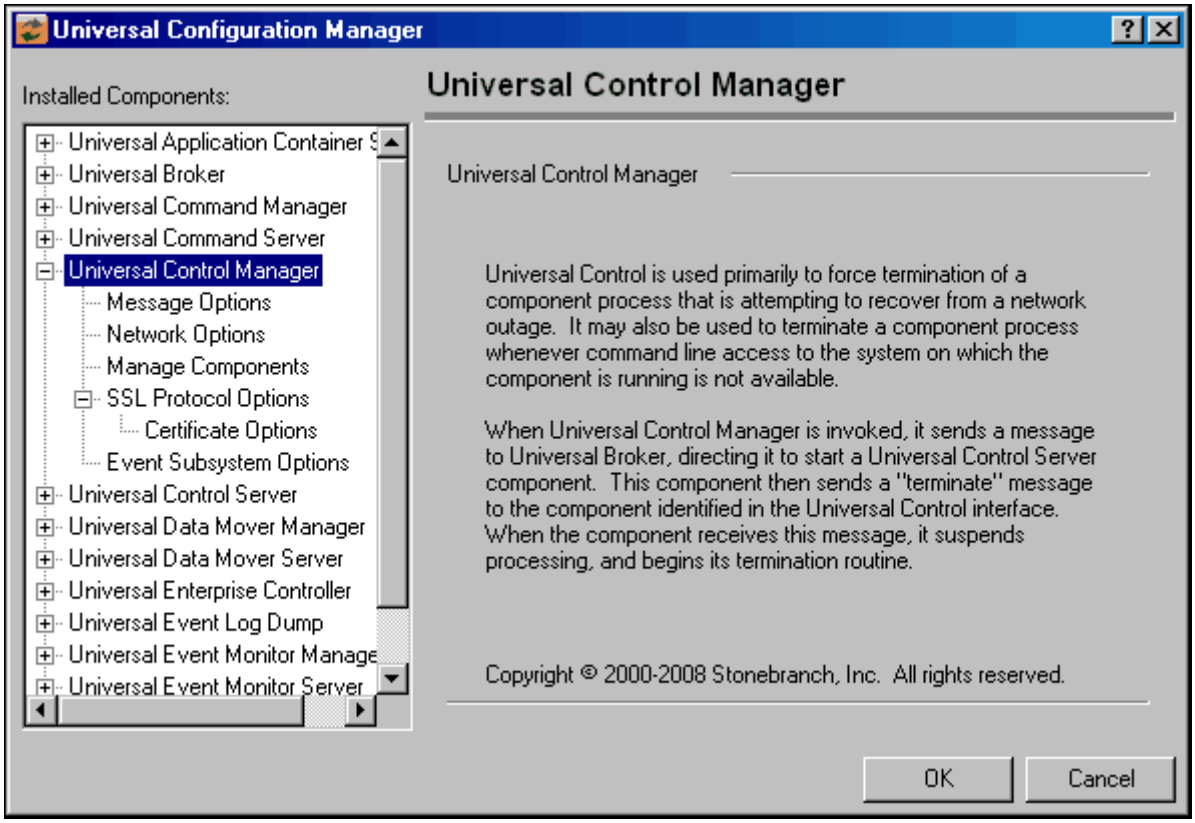

Figure 2.5 Universal Configuration Manager - Universal Control Manager

Universal Control Server

Figure 2.6 illustrates the Universal Configuration Manager screen for the Universal Control Server.

The Installed Components list identifies all of the Universal Control Server property pages.

The text describes the selected component, Universal Control Server.

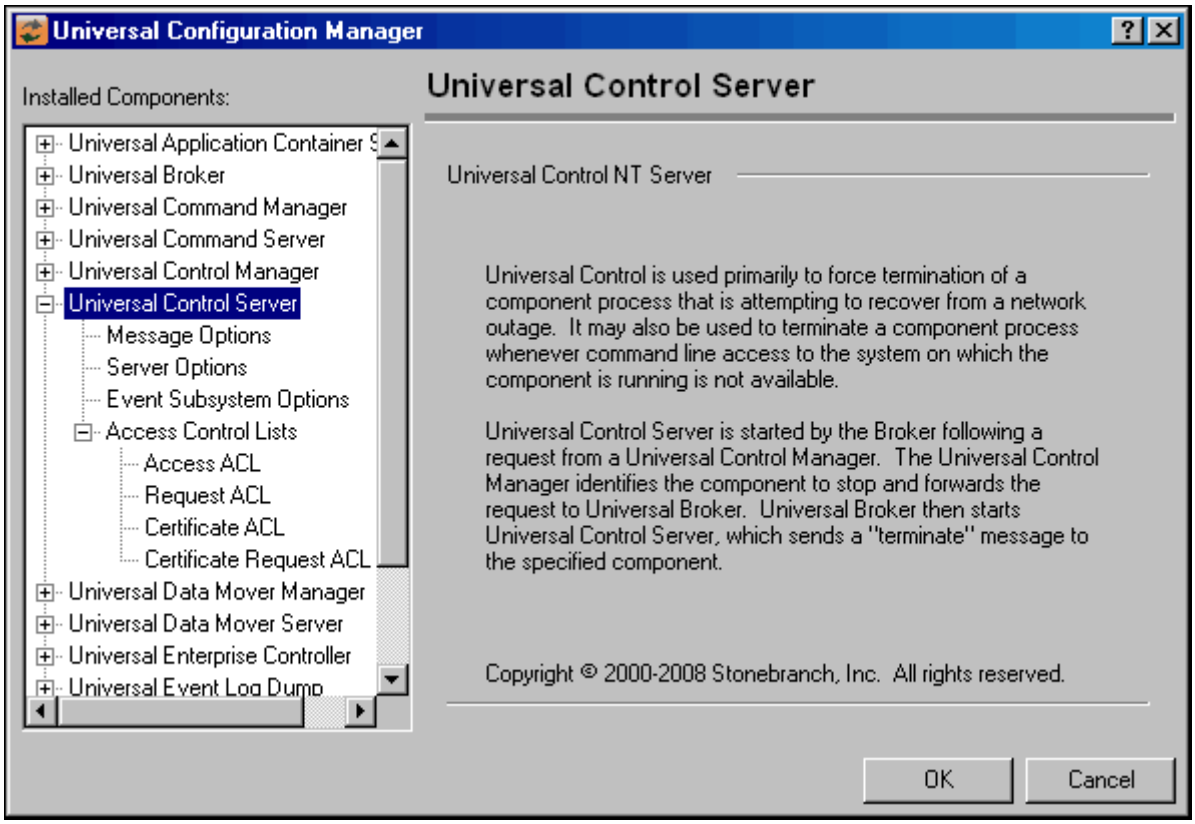

Figure 2.6 Universal Configuration Manager - Universal Control Server

Universal Event Log Dump

Figure 2.5 illustrates the Universal Configuration Manager screen for the Universal Event Log Dump utility.

The Installed Components list identifies all of the Universal Event Log Dump property pages.

The text describes the selected component, Universal Event Log Dump.

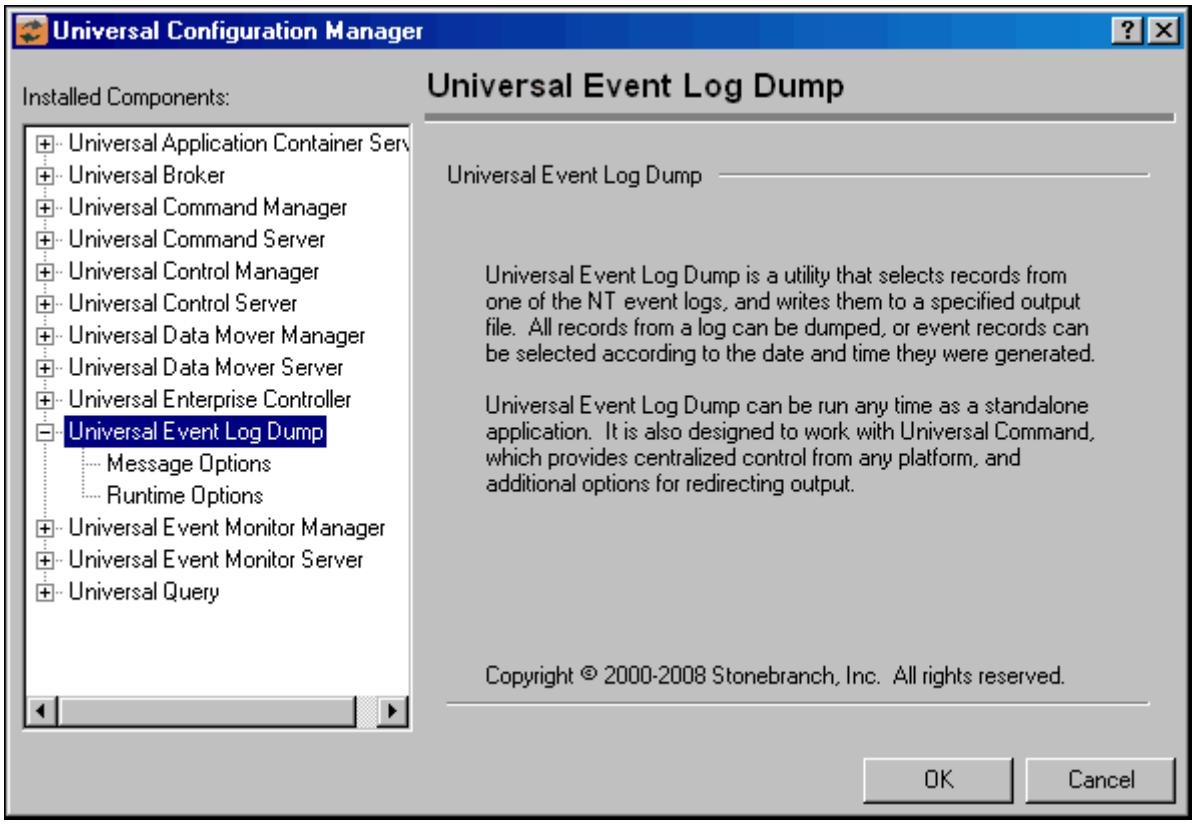

Figure 2.7 Universal Configuration Manager - Universal Event Log Dump

Universal Query

Figure 2.8 illustrates the Universal Configuration Manager screen for the Universal Query utility.

The Installed Components list identifies all of the Universal Query property pages.

The text describes the selected component, Universal Query.

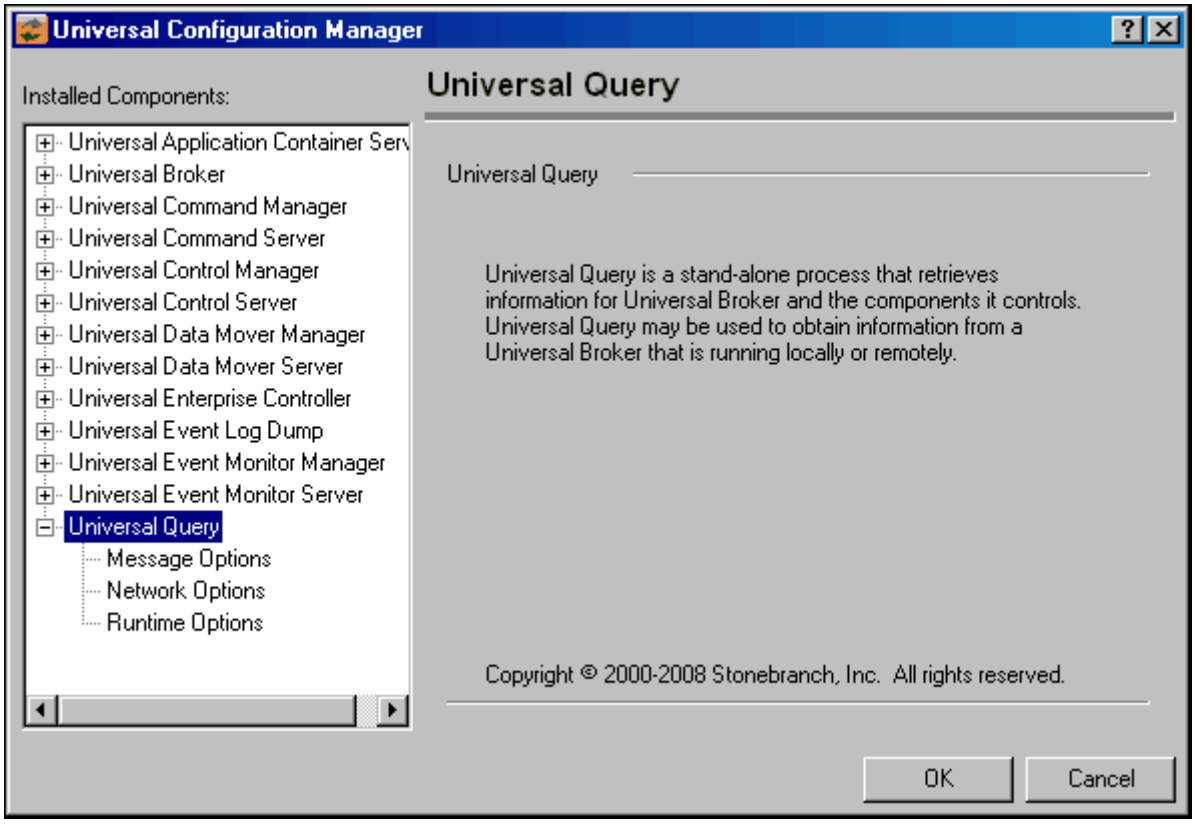

Figure 2.8 Universal Configuration Manager - Universal Query

2.5 Network Data Transmission

Distributed systems, such as Universal Command, communicate over data networks. All Stonebranch, Inc. Universal Products communicate using the TCP/IP protocol. The UDP protocol is not used for any product data communication over a network.

The Universal Products suite can utilize one of two network protocols:

1. Secure Socket Layer version 3 (SSLv3) provides the highest level of security available. SSL is a widely used and accepted network protocol for distributed software applications that are required to address all aspects of secure data transfer on private and public networks.
2. Universal Products version 2 (UNVv2) legacy protocol is provided for backward compatibility with previous versions of Universal Products.

The following sections discuss each of the protocols.

In addition to the network protocol used to transmit data, Universal Products application protocol is discussed as well.

2.5.1 Secure Socket Layer Protocol

Universal Products implement the SSL protocol using the OpenSSL library or the IBM z/OS System SSL library, available on the z/OS operating system. The most recent SSL standard is version3. A subsequent version was produced changing the name to Transport Layer Security version 1 (TLSv1). TLSv1 is the actual protocol used by Universal Products. TLSv1 is more commonly referred to simply as SSL and the term SSL is used throughout the rest of this documentation to mean TLSv1 unless otherwise noted.

The SSL protocol addresses the major challenges of communicating securely over a potentially insecure data network. The following sections discuss the issue of data privacy and integrity, and peer authentication.

Data Privacy and Integrity

People with sufficient technical knowledge and access to network resources can watch or capture data transmitting across the network. What they do with the data is up to them.

Data sent over the network that should remain private must be encrypted in a manner that unauthorized persons cannot determine what the original data contained regardless of their level of expertise, access to network resources, amount of data captured, and amount of time they have. The only party that should be able to read the data is the intended recipient.

As data is transmitted over the network, it passes through media and hardware of unknown quality that may erroneously change bits of data without warning. Additionally, although data may be encrypted, there is nothing stopping a malicious person from changing the data while it is transmitted over the network. The changed data may or may not be detected by the recipient depending on what changed and how it is processed. It may be accepted as valid data, but the information it represents is now erroneous

Data integrity must be protected from errors in transmission and malicious users. Data integrity checks insures that what was sent is exactly what is received by the recipient. Without integrity checks, there is no guarantee.

Encryption algorithms are used to encrypt data into an unreadable format. The encryption process is computationally expensive. There are a variety of encryption algorithms some of which perform better than others. Some algorithms offer a higher level of security than others. Typically, the higher level of security requires more computational resources.

Message digest algorithms are used to produce a Message Authentication Code (MAC) that uniquely identifies a block of data. The sender computes a MAC for the data being sent based on a shared secret key the sender and receiver hold. The sender sends the data and the MAC to the receiver. The receiver computes a new MAC for the received data based on the shared secret key. If the two MAC's are the same, data integrity is maintained, else the data is rejected as it has been modified. Message digest algorithms are often referred to as MAC's and can be used synonymously in most contexts.

The SSL standard defines a set of encryption and message digest algorithms referred to cipher suites that insure data privacy and data integrity. Cipher suites pair encryption algorithms with appropriate message digest algorithms. The two algorithms cannot be specified individually.

Universal Products supports a subset of the complete SSL cipher suites defined by the standard. The cipher suite name is formatted as an encryption algorithm abbreviation followed by the message digest algorithm abbreviation.

Table 2.2, below, identifies the supported cipher suites.

Cipher Suite Name	Description
RC4-SHA	128-bit RC4 encryption with SHA-1 message digest
RC4-MD5	128-bit RC4 encryption with MD5 message digest
AES256-SHA	256-bit AES encryption with SHA-1 message digest
AES128-SHA	128-bit AES encryption with SHA-1 message digest
DES-CBC3-SHA	128-bit Triple-DES encryption with SHA-1 message digest
DES-CBC-SHA	128-bit DES encryption with SHA-1 message digest
NULL-SHA	No encryption with SHA-1 message digest
NULL-MD5	No encryption with MD5 message digest

Table 2.2 Supported SSL cipher suites

Universal Products support one additional cipher suite name that is not part of the SSL protocol. The NULL-NULL cipher suite turns SSL off completely and instead uses the Universal Products Protocol (UNVv2) described below.

Peer Authentication

When communicating with a party across a data network, how do you ensure that the party you are communicating with (your peer) is who you believe? A common form of network attack is a malicious user representing themselves as another user or host.

Peer authentication insures that the peer is truly who they identify themselves as. Peer authentication applies to users, computer programs, and hardware systems.

SSL uses X.509 certificates and public and private keys to identify an entity. An entity may be a person, a program, or a system. A complete description of X.509 certificates is beyond the scope of this documentation. Section 2.8 X.509 Certificates provides an overview to help get the reader oriented to the concepts, terminology and benefits.

For additional details, the following web site is recommended:

http://www.faqs.org/rfcs/rfc3280.html

2.5.2 Universal Products Protocol

The Universal Products protocol (UNVv2) is a proprietary protocol that securely and efficiently transports data across data networks. UNVv2 is used in Universal Products prior to version 3 and will be available in future versions.

UNVv2 addresses data privacy and integrity. It does not address peer authentication.

Data Privacy and Integrity

Data privacy is ensured with data encryption algorithms. UNVv2 utilizes 128-bit RC4 encryption for all data encryption.

Data integrity is insured with message digest algorithms. UNVv2 utilizes 128-bit MD5 MAC's for data integrity. UNVv2 referred to data integrity as data authentication.

Encryption and integrity can be enabled and disabled on an individual basis.

Encryption keys are generated using a proprietary key agreement algorithm. A new key is created for each and every network session.

2.5.3 Universal Products Application Protocol

Universal Product components use an application-layer protocol to exchange data messages. The protocol has the following characteristics:

- Low-Overhead
- Secure
- Extensible
- Configurable Options

The following sections refer to two categories of data transmitted by Universal Products:

- Control data (or messages) consists of messages generated by Universal Products components in order to communicate with each other. The user of the product has no access to the control data itself.
- Application data (or messages) consists of data that is transmitted as part of the requested work being executed. For example, standard input and output data of jobs Universal Command executes. The data is created by the job and read or written by Universal Command on behalf of the job.

Low-Overhead

The protocol is lightweight, in order to minimize its use of network bandwidth. The product provides application data compression options, which reduces the amount of network data even further.

There are two possible compression methods:

- **`ZLIB`** method offers the highest compression ratios with highest CPU utilization.
- **`HASP`** method offers the lowest compression ratios with lowest CPU utilization.

Note: Control data is not compressed. Compression options are available for application data only.

Secure

The protocol is secure. All control data exchanged between Universal Products components are encrypted with a unique session key and contain a MAC. The encryption prevents anyone from analyzing the message data and attempting to circumvent product and customer policies. Each session uses a different encryption key to prevent "play back" types of network attacks, where messages captured from a previous session are replayed in a new session. This applies to both network protocols: SSL and UNVv2.

The security features used in the control messages are not optional. They cannot be turned off. The security features are optional for application data sent over the network.

The data encryption options affect the application data being sent over the network. Special fields, such as passwords, are always encrypted. The encryption option cannot be turned off for such data.

Extensible

The message protocol used between the Universal Products components is extensible. New message fields can be added with each new release without creating product component incompatibilities. This permits different component versions to communication with each other with no problems. This is a very important feature for distributed systems, since it is near impossible to upgrade hundreds of servers simultaneously.

New encryption and compression algorithms can be added in future releases without loosing backward compatibility with older releases. After a network connection is made, connection options are negotiated between the two Universal Products programs. The options negotiated include which encryption and compression algorithms are used for the session. Only algorithms that both programs implement are chosen in the negotiation process. The negotiation process permits two different program versions to communicate.

2.5.4 Configurable Options

The network protocol can be configured in ways that effect compress, encryption, code pages, and network delays.

The following configuration options are available on many of the Universal Products:

CODE_PAGE

The CODE_PAGE option specifies the code page translation table used to translate network data from and to the local code page for the system on which the program is executing.

A codepage table is text file that contain a two-column table. The table maps local single byte character codes to two-byte UNICODE character codes.

Code pages are located in the product National Language Support (NLS) directory or library. New code pages may be created and added to the NLS directory or library. The CODE_PAGE option value is simply the name of the code page file without any file name extension if present.

CTL_SSL_CIPHER_LIST

The CTL_SSL_CIPHER_LIST option specifies one or more SSL cipher suites that are acceptable to use for network communications on the control session, which is used for component internal communication.

The SSL protocol uses cipher suites to specify the combination of encryption and message digest algorithms used for a session. An ordered list of acceptable cipher suites can be specified in a most to least order of preference.

An example cipher suite list is RC4-MD5,RC4-SHA,AES128-SHA. The RC4-MD5 cipher suite is the most preferred and AES128-SHA is the least preferred.

When a manager and server first connect, they perform an SSL handshake. The handshake negotiates the cipher suite used for the session. The manager and server each have a cipher suite list and the first one in common is used for the session.

Why is a list of cipher suites helpful? A distributed software solution may cross many organizational and application boundaries each with their own security requirements. Instead of having to choose one cipher suite for all distributed components, the software components can be configured with their own list of acceptable cipher suites based on their local security requirements. When a high level of security is required, the higher CPU consuming cipher suite is justified. When lower level of security is acceptable, a lower CPU consuming cipher suite may be used. As long as the manager has both cipher suites in its list, it can negotiate either cipher suite with servers of different security levels.

DATA_AUTHENTICATION

The DATA_AUTHENTICATION option specifies whether or not the network data is authenticated. Data authentication verifies that the data did not change from the point it was sent to the point it was received.

Data authentication also is referred to as a data integrity in this document.

Data authentication occurs for each message sent over the network. If a message fails authentication, the network session is terminated and both programs end with an error.

The DATA_AUTHENTICATION option is applicable to the UNVv2 protocol only. SSL always performs authentication.

DATA_COMPRESSION

The DATA_COMPRESS option specifies that network data be compressed.

Compression attempts to reduce the amount of data to a form that can be decompressed to its original form. The compression ratio is the original size divided by the compressed size. The compression ratio value will depend on the type of data. Some data compress better than others.

Two methods of compression are available:

- ZLIB method provides the highest compression ratio with the highest use of CPU
- HASP method provides the lowest compress ratio with the lowest use of CPU.

Whether or not compression is used and which compression method is used depends on several items:

- Network bandwidth. If network bandwidth is small, compression may be worth the cost in CPU.
- CPU resources. If CPU is limited, the CPU cost may not be worth the reduced bandwidth usage.
- Data compression ratio. If the data does not compress well, it is probably not worth CPU cost. If the data ratio is high, the CPU cost may be worth it.

DATA_ENCRYPTION

The DATA_ENCRYPTION option specifies whether or not network data is encrypted.

Encryption translates data into a format that prevents the original data from being determined. Decryption translates encrypted data back into its original form.

The type of encryption performed depends on the network protocol being used, SSL or UNVv2.

Data encryption does increase CPU usage. Whether or not encryption is used depends on the sensitivity of the data and the security of the two host systems and the data network between the hosts.

DATA_SSL_CIPHER_LIST

The DATA_SSL_CIPHER LIST option specifies one or more SSL cipher suites that are acceptable to use for network communications on the data session, which is used for standard I/O file transmission.

(See CTL_SSL_CIPHER_LIST in this section.)

DEFAULT_CIPHER

The DEFAULT_CIPHER option specifies the SSL cipher suite to use (since SSL protocol requires a cipher suite) if the DATA_ENCRYPTION option is set to NO. The default DEFAULT_CIPHER is NULL-MD5 (no encryption, MD5 message digest).

All SSL cipher suites have a message digest for good reasons. The message digest ensures that the data sent are the data received. Without a message digest, it is possible for bits of the data packet to get changed without being noticed.

KEEPALIVE_INTERVAL

The KEEPALIVE_INTERVAL option specifies how often, in seconds, a keepalive message (also commonly known as a heartbeat message) is sent between a manager and server. A keepalive message ensures that the network and both programs are operating normally. Without a keepalive message, error conditions can arise that place one or both programs in an infinite wait.

A keepalive message is sent from the server to the manager. If the server does not receive a keepalive acknowledgement from the manager in a certain period of time (calculated as the maximum of 2 x NETWORK_DELAY or the KEEPALIVE_INTERVAL), the server considers the manager or network as unusable. How the server processes a keepalive time-out depends on what fault tolerant features are being used. If no fault tolerant features are being used, the server ends with an error. The manager expects to receive a keepalive message in a certain period of time (calculated as the KEEPALIVE_INTERVAL + 2 x NETWORK_DELAY.

NETWORK_DELAY

The NETWORK_DELAY option provides the ability to fine tune Universal product's network protocol. When a data packet is sent over a TCP/IP network, the time it takes to reach the other end depends on many factors, such as, network congestion, network bandwidth, and the network media type. If the packet is lost before reaching the other end, the other end may wait indefinitely for the expected data. In order to prevent this situation, Universal Products time out waiting for a packet to arrive in a specified period of time. The delay option specifies this period of time.

NETWORK_DELAY specifies the maximum acceptable delay in transmitting data between two programs. Should a data transmission take longer than the specified delay, the operation ends with a time out error. Universal Products will consider a time out error as a network fault.

The default NETWORK_DELAY value is 120 seconds. This value is reasonable for most networks and operational characteristics. If the value is too small, false network time outs could occur. If the value is too large, programs will wait a long period of time before reporting a time out problem.

SIO_MODE

The SIO_MODE option specifies whether the data transmitted over the network is processed as text data or binary data.

Text data is translated between the remote and local code pages. Additionally, end of line representations are converted

Text translation operates in two modes: direct and UCS. The default is direct. The direct translation mode exchanges code pages between Universal Products components to build direct translation tables. Direct translation is the fastest translation method when a significant amount (greater then 10K) of text data is transmitted. The code page exchange increases the amount of data sent over the network as part of the network connection negotiation. UCS translation does not require the exchange of code pages. For transactions that have little text data transmission, this is the fastest.

Binary data is transmitted without any data translation.

2.6 Universal Access Control List

Many Universal Products utilize the Universal Access Control List (UACL) feature as an extra layer of security to the services they offer. The UACL determines if a request is denied or allowed to continue and can assign security attributes to the request.

This section describes the UACL capabilities in general, non-component specific terms. See the appropriate component security sections for complete details on how a component utilizes the UACL feature.

The following Universal Product components use the UACL feature:

- Universal Broker uses UACLs to permit or deny TCP/IP connections based on the remote host IP address.

  See the Universal Broker 3.2.0 User Guide for complete details.
- Universal Command Server uses UACLs to permit or deny Manager access based on the Managers IP address and user ID, and to control whether or not the Manager request requires user authentication.

  See the Universal Command 3.2.0 User Guide for complete details.
- Universal Control Server uses UACLs to permit or deny Manager access based on the Managers IP address and user ID, and to control whether or not the Manager request requires user authentication.

  See the Universal Control chapter in this user guide for complete details.
- Universal Data Mover Server uses UACLs to permit or deny Manager access based on the Managers IP address and user ID.

  See the Universal Data Mover User Guide for complete details.

2.6.1 UACL Configuration

The method used to configure UACL rules is platform dependent. The following sections discuss each of the methods.

z/OS

All UACL rules are defined in library **UNVCONF**, member ACLCFG00. The Universal Broker allocates the UACL configuration data set to ddname **UNVACL**.

The UACL file syntax is the same as all other Universal Products z/OS configuration files. See Section 2.2.6 Configuration File Syntax for details.

UNIX

All UACL rules are defined in one file, **`uacl.conf`**. This file is required for products utilizing UACL rules; otherwise, the product will not start. The configuration file consists of zero or more UACL entries.

The UACL file syntax is the same as all other Universal Products UNIX configuration files. See Section 2.2.6 Configuration File Syntax for details.

Windows

All UACL rules are stored in the configuration file.

UACL entries for each component are maintained using the Universal Configuration Manager (see Section 2.4 Universal Configuration Manager).

OS/400

All UACL rules are defined in file **`unvconf`** and member **`uacl`**. This file is required for products utilizing UACL rules, else the product will not start. The configuration file consists of zero or more UACL entries.

The UACL file is searched for in the same manner as all other product configuration files. See Section 2.2.5 Configuration File for information on how configuration files are located.

The UACL file syntax is the same as all other Universal Products for OS/400 configuration files. See Section 2.2.6 Configuration File Syntax for details.

HP NonStop

All UACL rules are defined in one file, **`uaclcfg`**. This file is required for products utilizing UACL rules, else the product will not start. The configuration file consists of zero or more UACL entries.

The UACL file is located within the same subvolume as all other product configuration files.

The UACL file syntax is the same as all other Universal Products HP NonStop configuration files. See Section 2.2.6 Configuration File Syntax for details.

2.6.2 UACL Entries

UACL entries are composed of two parts: type and rule.

- Type identifies the Universal Products component for which the rule applies. For example, the Universal Broker product utilizes UACL rules of type **`ubroker_access`**.
- Rule defines the client's identity and the client's request for which the entry pertains and the security attributes it enforces.

UACL configuration file syntax is the same as all other configuration files, where the configuration file keyword corresponds to the UACL type part and the configuration file value corresponds to the UACL rule part.

The entire rule part of the UACL entry must be enclosed in quotation characters, not just a sub-field of the rule, if a space or tab is part of the value.

The correct syntax would be as follows:

```
uctl_request "prod.host.name,MVS USER,user,cmd,DSPLIB
QGPL,allow,auth"
```

For each client that connects and sends a request, Broker and Server components search UACL entries to find the best match for the client identity and the client request. Entries are searched in the order they are listed. The first entry found stops the search.

Note: There is no limit to the number of UACL entries that can be specified.

Client Identification

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 and reported user account.

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.

Table 2.3, below, 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 (`,`) 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 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 an RDN value is case sensitive or not 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.
IP ADDRESS	Matches the X.509 **iPAddress** field of the **subjectAltName** extension value. An example IPADDRESS value is: • **ipaddress=10.20.30.40**
SERIAL NUMBER	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**

Table 2.3 Certificate Map Matching Criteria

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.

Table 2.4, below, defines the certificate identifier field as used in UACL entries.

Criteria	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**.

Table 2.4 Certificate Identifier Field

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. There are exceptions to this though. 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 Product 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.

Table 2.5, below, defined possible matching criteria for IP address and user account client identification.

Criteria	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 **131.155.73.255**. 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. • 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**.

Table 2.5 Client IP Address - Matching Criteria

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 is dependent on the component's UACL entry type.

Table 2.6, below, lists a complete set of the request fields that are possible. See each component's UACL entry definitions for further details.

Criteria	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.
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**.

Criteria	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**.

Table 2.6 Request Fields

Certificate Based and Non Certificate Based UACL Entries

Universal Products 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, **cmd_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.

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

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

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.

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

2.7 Message and Audit Facilities

All Universal Products have the same message facilities. Messages - in this context - are text messages written to a console, file, or system log that:

1. Document the actions taken by a program.
2. Inform users of error conditions encountered by a program.

This section describes the message and audit facilities that are common to all Universal Products. (See the individual Universal Product documentation for additional details.)

2.7.1 Message Types

There are six types (or severity levels) of Universal Products messages. (The severity level is based on the type of information provided by those messages.)

1. Audit messages document the configuration options used by the program's execution and resource allocation details. They provide complete description of the program execution for auditing and problem resolution.
2. Informational messages document the actions being taken by a program. They help determine the current stage of processing for a program. Informational messages also document statistics about data processed.
3. Warning messages document unexpected behavior that may cause or indicate a problem.
4. Error messages document program errors. They provide diagnostic data to help identify the cause of the problem.
5. Diagnostic messages document diagnostic information for problem resolution.
6. Alert messages document a notification that a communications issue, which does not disrupt the program or require action, has occurred.

The MESSAGE_LEVEL configuration option in each Universal Product component lets you specify which messages are written (see Section 2.7.3 Message Levels).

2.7.2 Message ID

Each message is prefixed with a message ID that identifies the message.

The message ID format is **UNVnnnnl**, where:

- **nnnn** is the message number.
- **l** is the message severity level:
  - **A** (Audit)
  - **I** (Informational)
  - **W** (Warning)
  - **E** (Error)
  - **T** (alerT)
  - **D** (Diagnostic)

Note: The Universal Products 3.2.0 Messages and Codes document identifies all messages numerically, by product, using the **nnnn** message number.

2.7.3 Message Levels

Each Universal Product includes a MESSAGE_LEVEL configuration option that lets you select which levels (that is, severity levels) of messages are to be written.

- *Audit* specifies that all audit, informational, warning, and error messages are to be written.
- *Informational* specifies that all informational, warning, and error messages are to be written.
- *Warning* specifies that all warning and error messages are to be written.
- *Error* specifies that all error messages are to be written.
- *Trace* specifies that a trace file is created, to which data used for program analysis will be written. The trace file name and location are Universal Product dependent (see the appropriate Universal Product documentation for details).

  (Trace should be used only at the request of Stonebranch, Inc. Customer Support.)

Note: Diagnostic and Alert messages always are written, regardless of the level selected in the MESSAGE_LEVEL option.

2.7.4 Message Destinations

The location to which messages are written is the message destination.

Some Universal Products have a MESSAGE_DESTINATION configuration option that specifies the message destination. If a program is used only from the command line or batch job, it may have only one message destination, such as standard error. Valid destination values will depend on the host operating system.

z/OS

Universal Products on z/OS run as batch jobs or started tasks. Batch jobs do not provide the MESSAGE_DESTINATION option. All messages are written to the SYSOUT ddname.

Started task message destinations are listed in the table below.

Destination	Description
LOGFILE	Messages are written to ddname UNVLOG. All messages written to log files include a date and time stamp and the program's USS process ID.
SYSTEM	Messages are written to the console log as WTO messages.

UNIX

Message destinations are listed in the table below.

Destination	Description
STDERR	Messages are written to standard error. This destination is most useful for console commands.
LOGFILE	Messages are written to a log file. Not all programs provide this destination. The recommended directory for log files is `/var/opt/universal/log`. This can be changed with the LOG_DIRECTORY option. All messages written to log files include a date and time stamp and the program's process ID.
SYSTEM	Messages are written to the syslog daemon. Not all programs provide this destination. Universal programs that execute as daemons write to the syslog's daemon facility. All messages include the programs process ID. If an error occurs writing to the syslog, the message is written to the system console.

Windows

Message destinations are listed in the table below.

Destination	Description
STDERR	Messages are written to standard error. This destination is most useful for console commands.
LOGFILE	Messages are written to a log file. Not all programs provide this destination. Log files are written to product specific log directories, which can be modified with the LOG_DIRECTORY option. All messages written to log files include a date and time stamp and the program's process ID.
SYSTEM	Messages are written to the Windows Application Event Log.

OS/400

Message destinations are listed in the table below.

Destination	Description
STDERR	Messages are written to standard error. A batch job's standard error file is allocated to the print file QPRINT.
LOGFILE	Messages are written to the job's job log.
SYSTEM	Messages are written to the system operator message queue QSYSOPR.

HP NonStop

Message destinations are listed in the table below.

Destination	Description
STDERR	Messages are written to standard error.
LOGFILE	Messages are written to a log file. Not all programs provide this destination. Log files are written the **$SYSTEM.UNVLOG** subvolume. All messages written to log files include a date and time stamp and the program's process ID.

2.8 X.509 Certificates

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 someone wants to send you a secret message, they 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 message, such as a request for $100,000 purchase, you can "sign" it with your private key.

Note: 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.

2.8.1 Sample Certificate Directory

Figure 2.9, below, provides a sample diagram of a small X.500 directory.

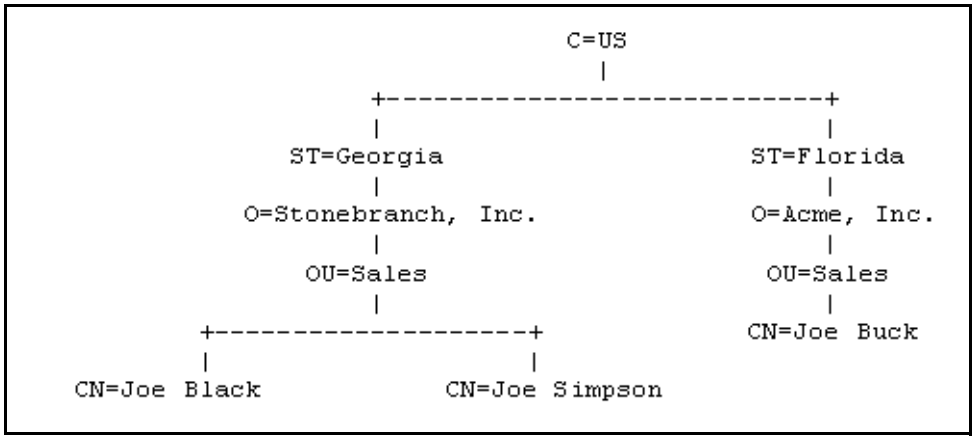

Figure 2.9 X.500 Directory (Sample)

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.

2.8.2 Sample X.509 Certificate

Figure 2.10, below, 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 2004 GMT
            Not After : Aug 20 12:59:55 2005 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:
                    77:13:75:57:08:29:45:29: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:75:8d: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:73:45:66: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:38:73:d2:85:a3:c7:24:21:47:ee:f2:
        8c:0d
```

Figure 2.10 X.509 Version3 Certificate (Sample)

Note: The contents of a certificate file does not look like the information in Figure 2.10, which is produced by a certificate utility using the certificate file as input. Certificates can be saved in multiple file formats, so their file contents will look very different.

Certificate Fields

A certificate is composed of many fields.

Figure 2.11, below, describes the main 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.

Figure 2.11 Certificate Fields

2.8.3 SSL Peer Authentication

The SSL 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 Broker.

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

Certificate Verification

The Universal Product 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.

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. Therefore, when 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 Products that utilize certificates.

Certificate Identification

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

A Universal Product Manager validates a Broker certificate by the Broker host name or 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.

Should certificate identification fail, 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.

Certificate Support

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

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

Although implementing a full featured PKI infrastructure is beyond the scope of Universal Products and this documentation, some assistance is provided using the OpenSSL toolkit (http://www.openssl.org).

Universal Products on most of the supported platforms utilize the OpenSSL toolkit for its SSL and certificate implementation. OpenSSL is delivered on most UNIX distributions and Windows distributions are available on the OpenSSL web site.

Universal Products 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 Product suite includes an X.509 certificate utility, Universal Certificate, to create certificates for use in the Universal Product suite. See the Universal Certificate chapter in the Universal Products Utilities 3.2.0 User Guide for details.

Chapter 3
Universal Certificate

3.1 Overview

The Universal Certificate (UCERT) utility creates digital certificates and private keys, which Universal Product programs can use to securely identify users and computer systems.

The certificates created by Universal Certificate comply with the *Internet X.509 Public Key Infrastructure* RFC 3280 document; however, not all certificate fields are supported.

The aim of Universal Certificate is to provide a simple certificate creation utility to be used if no Public Key Infrastructure (PKI) is available in your company. It is not a replacement for a corporate PKI.

See Section 2.8 X.509 Certificates in Chapter 2 Features for an introduction to X.509 certificates and how they are used by Universal Product components.

3.2 Usage

Universal Certificate performs the following operations, as specified by command line configuration options:

- Create certificates, certificate requests, private keys, certificate revocation lists (CRLs), and PKCS#12-encoded transport files.
- Print certificates, certificate requests, CRLs, and PKCS#12-encoded transport files.
- Verify certificates

The following sections describe each of these operations.

3.2.1 Certificate

A certificate is an electronic object use for identification purposes. A certificate identifies a person or computer system, as well as the party that issued the certificate. Certificates are issued by Certificate Authorities (CAs). A certificate only can be trusted if the CA that issued the certificate is trusted.

A certificate is created using the following input:

- Certificate request: Identifies the person / computer system for which the certificate is to be issued.
- CA certificate Identifies the Certificate Authority (CA) that is issuing the certificate.
- CA private key Signs (digitally) the certificate.

3.2.2 Certificate Requests

A certificate request is a request for a CA to issue a certificate. A certificate request contains all of the information required to identify a user / computer system.

The certificate request is saved in a file that is sent to a CA. The CA is responsible for verifying the information in the request and creating the final certificate based on that information.

When a certificate request is created, its corresponding private key also is created. The private key is written to a file and must remain private. File system security must be used to prevent unauthorized access to the private key file. Additionally, the private key can be protected with a password.

Certificate requests are encoded in Public-Key Cryptography Standards (PKCS) #10 syntax. Private keys are encoded in PKCS #8 syntax.

3.2.3 Certificate Revocation List

A Certificate Revocation List (CRL) is created by the Certificate Authority (CA). The list includes all certificates issued by the CA that subsequently have been revoked by the CA for some reason. The CRL is signed by the issuing CA.

A CRL is used as part of the certificate verification process to ensure that a certificate still is valid.

3.2.4 Transport Files

A transport file is a PKCS #12-encoded file generated by Universal Certificate in order to securely transfer a user’s certificate and private key across systems.

Many applications that manage digital certificates - including RACF on z/OS and the Certificate Management add-in for the Microsoft Management Console application on Windows - can import a user's certificate using a transport file.

Universal Certificate also can extract certificate and private key information from PKCS #12-encoded transport files created by other applications. Command line options allow this extracted information to be stored in local files. If a CA's certificate - or the CA certificate chain - was added to the transport file, Universal Certificate can extract it as well.

3.2.5 Printing

Certificates, certificate requests, and transport files are saved in encoded files that are not easily readable (by people). However, they can be printed in text format.

3.2.6 Verification

Certificate verification is the process of verifying that a certificate is valid.

The certificate process consists of

1. Verifying that the certificate is issued by a trusted CA.
2. Verifying that the certificate is not revoked by the CA.

3.2.7 File Formats

Certificates, certificate requests, and private keys are stored in files.

The following file formats are supported:

- Privacy Enhanced Mail (PEM)
  PEM is the format described in RFCs 1421-1424. PEM is a base64 encoding with header and trailer lines added to identify the contents. PEM is a text format suitable for email and text file transfers.
- Distinguished Encoding Rules (DER)
  DER is an encoding rule based on the Abstract Syntax Notation 1 (ASN.1) specification. DER is a binary file format. When transferred across a network, it must be transferred in a binary or image mode.

All certificates and keys are encoded in an ASN.1 format. The PEM format is a text representation of the DER format.

Note: Universal Certificate supports only the DER format for PKCS #12-encoded transport files.

z/OS

PEM- and DER-formatted files can be written either to a member of a partitioned data set or a sequential data set. The record format must be variable or variable blocked. The record length must be at least 80.

3.2.8 Universal Certificate Database

Universal Certificate uses a database to maintain issued and revoked certificates. The database is required for certificate creation, certificate revocation, and CRL creation.

There is a one-to-one correspondence between a CA and a certificate database. That is, a unique database must be used for each CA, and each CA should use only one database.

The database is a very important element in maintaining a CA. Consequentially, it must be properly managed. The database must be secured from unauthorized updates and routinely backed up. The database is a regular text file.

z/OS

The database is allocated to ddname **UNVDB**. The database allocation attributes are DSORG=PS, RECFM={FB | F}, and LRECL=1120. The block size must be a multiple of LRECL if RECFM is FB.

UNIX and Windows

The database file name is specified with the CERT_DB option. If CERT_DB is not used, the database is created in the current working directory with name **`ucert.db`**.

3.3 Configuration Options

Table 3.1, below, identifies the Universal Certificate for configuration options for the UNIX, Windows, and z/OS operating systems.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CA	Specification for whether or not the certificate should be marked as a Certificate Authority certificate.
CA_CERT_FILE	Name from which the CA certificate is read.
CA_CERT_FORMAT	Format of the CA certificate file specified by CA_CERT_FILE.
CERT_DB	Certificate database name.
CERT_FILE	File name to which the certificate is written.
CERT_FORMAT	Format of the certificate file specified by CERT_FILE.
CODE_PAGE	Character code page used to translate text data.
COMMAND_FILE_ENCRYPTED	Name of an encrypted command file.
COMMAND_FILE_PLAIN	Name of a plain text command file.
COMMON_NAME	Common name of the **subject** field of a certificate.
COUNTRY	Country name of the **subject** field of a certificate.
CREATE	Specification that UCERT is to create a certificate request or a certificate.
CRL_FILE	File name to which the Certificate Revocation List (CRL) is written.
CRL_FORMAT	Format of the CRL file specified by CRL_FILE.
DNS_NAME	Domain Name System (DNS) name of the computer system for which the certificate identifies.
EMAIL_ADDRESS	Email address of the entity identified by the certificate.
ENCRYPTION_KEY	Key used to encrypt the command file.
HELP	Writes a description of the command options and their format.
IP_ADDRESS	Internet Protocol (IP) address of the computer system for which the certificate identifies.
KEY_SIZE	Key size of the RSA public / private keys.
LOCALITY	Locality name of the **subject** field of a certificate.
MESSAGE_LEVEL	Level of messages to write.
NEXT_UPDATE_DAYS	Number of days to the next CRL update.
NEXT_UPDATE_HOURS	Number of hours to the next CRL update.
NLS_DIRECTORY	Directory name where the code page UTT files are located.
NOT_AFTER_DATE	Last day for which the certificate is considered valid.
NOT_BEFORE_DATE	First day for which the certificate is considered valid.
ORGANIZATION	Organization name of the **subject** field of a certificate.
ORGANIZATIONAL_UNIT	Organizational unit name of the **subject** field of a certificate.

Option Name	Description
PRINT	Specification that UCERT is to print a certificate request or a certificate.
PRIVATE_KEY_FILE	File name from which the RSA private key is read or to which the RSA private key is written.
PRIVATE_KEY_FORMAT	Format of the private key file specified by PRIVATE_KEY_FILE.
PRIVATE_KEY_PWD	Password used to read and write the private key file specified by PRIVATE_KEY_FILE.
REQUEST_FILE	File name from which the certificate request is read or to which the certificate request is written.
REQUEST_FORMAT	Format of the certificate request file specified by REQUEST_FILE.
REVOKE	Specification that UCERT is to revoke a certificate.
REVOKE_REASON	Reason a certificate is being revoked.
SERIAL_NUMBER	Unique serial number to be assigned to the created certificate.
STATE	State name of the **`subject`** field of a certificate.
TRANSPORT_FILE	File containing certificate / private key information.
TRANSPORT_FILE_PWD	Password used to protect the file specified by TRANSPORT_FILE.
VERIFY	Specification that UCERT is to verify a certificate.
VERSION	Prints the program version and copyright information.

Table 3.1 Universal Certificate Configuration Options - z/OS, UNIX, and Windows

3.4 Universal Certificate for z/OS

Universal Certificate for z/OS executes as a batch job.

This section describes the Universal Certificate for z/OS JCL and command line options.

3.4.1 JCL Procedure

Figure 3.1, below, illustrates the Universal Certificate for z/OS JCL procedure (**UCRPRC**, located in the **SUNVSAMP** library), that is provide to simplify the execution JCl and future maintenance.

```
//UCRPRC   PROC UPARM=,               -- UCERT options
//             UCRPRE=#SHLQ.UNV,
//             UCRDBPRE=#PHLQ.UNV
//*
//PS1      EXEC PGM=UCERT,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DSN=&UCRPRE..SUNVLOAD,
//             DISP=SHR
//*
//UNVDB    DD  DSN=&UCRDBPRE..UCRDB,
//             DISP=SHR
//UNVNLS   DD  DSN=&UCRPRE..SUNVNLS,
//             DISP=SHR
//UNVTRACE DD  SYSOUT=*
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

Figure 3.1 Universal Certificate for z/OS – JCL Procedure

3.4.2 DD Statements used in JCL Procedure

Table 3.2, below, describes the DD statements used in the Universal Certificate for z/OS JCL illustrated in Figure 3.1.

ddname	Description
STEPLIB	Load library in which program UCERT is located.
UNVDB	UCERT certificate database.
UNVNLS	UCERT national language support ddname.
UNVTRACE	UCERT trace ddname.
SYSPRINT	UCERT standard output ddname.
SYSOUT	UCERT standard error ddname.

Table 3.2 Universal Certificate for z/OS – DD Statements in JCL

3.4.3 JCL

Figure 3.2, below, illustrates the Universal Certificate for z/OS JCL.

```
//UCERT    EXEC PGM=UCERT
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//UNVNLS   DD  DISP=SHR,DSN=UNV.SUNVNLS
//UNVDB    DD  DISP=SHR,DSN=UNV.UCRDB

//UNVTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSIN    DD  DUMMY
```

Figure 3.2 Universal Certificate for z/OS – JCL

3.4.4 Command Line Syntax

Figure 3.3, Figure 3.4, and Figure 3.5, below, illustrate the syntax – using the long form of command line options – of Universal Certificate for z/OS.

```
ucert
[-codepage codepage]
[-level {trace|audit|info|warn|error}]
[ -file ddname | -encryptedfile ddname [-key key] ]

Creating a certificate request.
{-create request
-request_file ddname [-request_format {pem|der}]
-private_key_file ddname [-private_key_format {pem|der}]
[-private_key_pwd password]
[-key_size {512|1024|2048}]
[-country name]
[-state name]
[-locality name]
[-organization name]
[-organizational_unit name]
[-common_name name]
{ [-dns_name name] | [-ip_address name] }
[-email_address name]

Creating a certificate from a certificate request.
| -create cert
-request_file ddname [-request_format {pem|der}]
-cert_file ddname [-cert_format {pem|der}]
-private_key_file ddname [-private_key_format {pem|der}]
[-private_key_pwd password]
-ca_cert_file ddname [-ca_cert_format {pem|der}]
[-serial_number number]
[-not_before_date date] [-not_after_date date]
[-ca {yes|no}]
[-cert_db ddname]
```

Figure 3.3 Universal Certificate for z/OS - Command Line Syntax (1 of 3)

```
Creating a certificate from a transport file.
| -create cert
-transport_file ddname [-transport_file_pwd password]
-cert_file ddname [-cert_format {pem|der}]
-private_key_file ddname [-private_key_format {pem|der}]
-ca_cert_file ddname [-ca_cert_format {pem|der}]

Creating a certificate revocation list.
| -create crl
-crl_file ddname [-crl_format {pem|der}]
-ca_cert_file ddname [-ca_cert_format {pem|der}]
-private_key_file ddname [-private_key_format {pem|der}]
[-private_key_pwd password]
-next_update_days days
-next_update_hours hours
[-cert_db ddname]

Creating a transport file.
| -create transport
-transport_file ddname [-transport_file_pwd password]
-cert_file ddname [-cert_format {pem|der}]
-private_key_file ddname [-private_key_format {pem|der}]
[-private_key_pwd password]
-ca_cert_file ddname [-ca_cert_format {pem|der}]

Revoking a certificate.
| -revoke cert
[-revoke_reason {unspecified|keyCompromise|caCompromised|
affiliationChange|superseded|cessationofOperation|privelegeWithdrawn}]
-cert_file ddname [-cert_format {pem|der}]
[-cert_db ddname]

Printing a certificate request.
| -print request
-request_file ddname [-request_format {pem|der}]

Printing a certificate.
| -print cert
-cert_file ddname [-cert_format {pem|der}]
```

Figure 3.4 Universal Certificate for z/OS - Command Line Syntax (2 of 3)

```
Printing a certificate revocation list.
| -print crl
-crl_file ddname [-crl_format {pem|der}]

Printing a transport file.
| -print transport
-transport_file ddname [-transport_file_pwd password]

Verifying a certificate.
| -verify cert
-cert_file ddname [-cert_format {pem|der}]
-ca_cert_file ddname [-ca_cert_format {pem|der}]
-crl_file ddname [-crl_format {pem|der}]

}

ucert
{ -help | -version }
```

Figure 3.5 Universal Certificate for z/OS - Command Line Syntax (3 of 3)

3.5 Universal Certificate for UNIX and Windows

Universal Certificate for UNIX and Windows executes as a command line application.

This section describes the command line syntax of Universal Certificate for UNIX and Windows.

3.5.1 Command Line Syntax

Figure 3.6, Figure 3.7, and Figure 3.8, below, illustrate the syntax – using the long form of command line options – of Universal Certificate for UNIX and Windows.

```
ucert
[-codepage codepage]
[-nls_directory directory]
[-level {trace|audit|info|warn|error}]
[ -file filename | -encryptedfile filename [-key key] ]

Creating a certificate request.
{-create request
-request_file file [-request_format {pem|der}]
-private_key_file file [-private_key_format {pem|der}]
[-private_key_pwd password]
[-key_size {512|1024|2048}]
[-country name]
[-state name]
[-locality name]
[-organization name]
[-organizational_unit name]
[-common_name name]
{ [-dns_name name] | [-ip_address name] }
[-email_address name]
```

Figure 3.6 Universal Certificate for UNIX and Windows - Command Line Syntax (1 of 3)

```
Creating a certificate from a certificate request.
| -create cert
-request_file file [-request_format {pem|der}]
-cert_file file [-cert_format {pem|der}]
-private_key_file file [-private_key_format {pem|der}]
[-private_key_pwd password]
-ca_cert_file file [-ca_cert_format {pem|der}]
[-serial_number number]
[-not_before_date date] [-not_after_date date]
[-ca {yes|no}]
[-cert_db file]

Creating a certificate from a transport file.
| -create cert
-transport_file file [-transport_file_pwd password]
-cert_file file [-cert_format {pem|der}]
-private_key_file file [-private_key_format {pem|der}]
-ca_cert_file file [-ca_cert_format {pem|der}]

Creating a certificate revocation list.
| -create crl
-crl_file file [-crl_format {pem|der}]
-ca_cert_file file [-ca_cert_format {pem|der}]
-private_key_file file [-private_key_format {pem|der}]
[-private_key_pwd password]
-next_update_days days
-next_update_hours hours
[-cert_db file]

Creating a transport file.
| -create transport
-transport_file file [-transport_file_pwd password]
-cert_file file [-cert_format {pem|der}]
-private_key_file file [-private_key_format {pem|der}]
[-private_key_pwd password]
-ca_cert_file file [-ca_cert_format {pem|der}]
```

Figure 3.7 Universal Certificate for UNIX and Windows - Command Line Syntax (2 of 3)

```
Revoking a certificate.
| -revoke cert
[-revoke_reason {unspecified|keyCompromise|caCompromised|
affiliationChange|superseded|cessationofOperation|privelegeWithdrawn}]
-cert_file file [-cert_format {pem|der}]
[-cert_db file]

Printing a certificate request.
| -print request
-request_file file [-request_format {pem|der}]

Printing a certificate.
| -print cert
-cert_file file [-cert_format {pem|der}]

Printing a certificate revocation list.
| -print crl
-crl_file file [-crl_format {pem|der}]

Printing a transport file.
| -print transport
-transport_file file [-transport_file_pwd password]

Verifying a certificate.
| -verify cert
-cert_file file [-cert_format {pem|der}]
-ca_cert_file file [-ca_cert_format {pem|der}]
-crl_file file [-crl_format {pem|der}]

}

ucert
{ -help | -version }
```

Figure 3.8 Universal Certificate for UNIX and Windows - Command Line Syntax (3 of 3)

3.6 Examples of Universal Certificate

This section 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.

The following list provides a link to each example.

- Creating a Certificate Authority Certificate
- Creating a Certificate

3.6.1 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 command creates:

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

```
-c request –r req.pem –e 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 command creates the CA certificate and writes it to file **cacert.pem**.

```
-c cert –r req.pem –d cacert.pem –e 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.

3.6.2 Creating a Certificate

There are two steps in creating a certificate:

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

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 command creates:

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

```
-c request –r req.pem –e pkey.pem –country US –state Maryland
–locality Baltimore –organization “Acme, Inc.” –common_name “Joe Buck”
```

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 command creates the certificate and writes it to file **cert.pem**.

```
-c cert –r req.pem –d cert.pem –e cakey.pem –a cacert.pem
```

Chapter 4
Universal Control

4.1 Overview

This chapter provides information on the Universal Control (UCTL) utility.

Universal Control consists of two components:

- UCTL Manager
- UCTL Server

The UCTL Manager executes one of three different control requests, which is specified via a corresponding configuration option:

1. Start (START_CMD option)
   Starts a component on the specified system.
2. Stop (STOP_CMD option)
   Stops a component on the specified system.
3. Refresh (REFRESH_CMD option)
   Directs Universal Broker on the remote system to refresh the configuration data of all components, including itself, or a single component (currently, only Universal Event Monitor Server).

One of these control requests must be specified for each execution of Universal Control Manager. Additional input (required and optional) to each execution of the UCTL Manager is made via additional configuration options, which control product behavior and resource allocation for that execution.

Upon execution, UCTL Manager connects to the UCTL Server and processes the request. UCTL Manager registers with a locally running Universal Broker. Consequentially, a Universal Broker must be running in order for a UCTL Manager to execute.

4.2 Universal Control Manager for z/OS

This chapter provides information on Universal Control (UCTL) Manager specific to the z/OS operating system.

UCTL Manager for z/OS executes as a batch job.

4.2.1 Usage

This section describes the control requests, JCL procedure and JCL, configuration and configuration options, and command line syntax of UCTL Manager for z/OS.

Control Requests

UCTL Manager for z/OS supports all three Universal Control control requests:

1. Start (START_CMD option)
2. Stop (STOP_CMD option)
3. Refresh (REFRESH_CMD option)

Section 4.2.2 Examples of UCTL Manager for z/OS provides an example of each request.

JCL Procedure

Figure 4.1, below, identifies the UCTL Manager for z/OS JCL procedure (**UCTLPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//UCTLPRC  PROC UPARM=,               -- UCTL options
//             UCMDPRE=#SHLQ.UNV
//*
//PS1      EXEC PGM=UCTL,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UCMDPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UCMDPRE..SUNVNLS
//UNVTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
```

Figure 4.1 Universal Control Manager for z/OS – JCL Procedure

DD Statements used in JCL Procedure

Table 4.1, below, describes the DD statements used in the UCTL Manager for z/OS JCL procedure illustrated in Figure 4.1.

ddname	DCB Attributes *	Mode	Description
STEPLIB	DSORG=PO, RECFM=U	Input	Load library containing the program being executed.
UNVNLS	DSORG=PO, RECFM=(F, FB, V, VB)	Input	UCTL national language support library. Contains message catalogs and code page translation tables.
UNVTRACE	DSORG=PS, RECFM=(F, FB, V, VB)	Output	UCTL trace output.
SYSPRINT	DSORG=PS, RECFM=(F, FB, V, VB)	Output	stdout file for the UCTL program. UCTL does not write any messages to SYSPRINT.
SYSOUT *	DSORG=PS, RECFM=(F, FB, V, VB)	Output	stderr file for the UCTL program. UCTL writes its messages to SYSOUT.
* The C runtime library determines the default DCB attributes. Refer to the IBM manual *OS/390 C/C++ Programming Guide* for details on default DCB attributes for stream I/O			

Table 4.1 Universal Control Manager for z/OS – DD Statements in JCL Procedure

JCL

Figure 4.2, below, illustrates the UCTL Manager for z/OS JCL using the **UCTLPRC** procedure illustrated in Figure 4.1.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UCTLPRC
//SYSIN    DD  *
 -stop 10312932 -host dallas -userid joe -pwd akkSdiq
/*
```

Figure 4.2 Universal Control Manager for z/OS – JCL

Job step STEP1 executes **UCTLPRC**.

The configuration options are specified on the SYSIN DD.

Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCTL Manager.
- Setting options and preferences for a single execution of UCTL Manager.

Configuration options are read from the following sources:

1. PARM keyword
2. SYSIN ddname
3. Command file ddname
4. Configuration file

The order of precedence is the same as the list above; command line being the highest, and configuration file being the lowest. That is, options specified via a PARM keyword override options specified via a SYSIN ddname, and so on.

The UCTL Manager configuration file is provided to the manager by the local Universal Broker with which it registers. The UCTL Manager configuration file is located in the **UCTCFG00** member of the PDSE allocated to the **UNVCONF** ddname in the Universal Broker started task.

The configuration file, provided by the local Universal Broker, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCTL Manager.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Note: For any changes to the UCTL Manager configuration file to become active, a Universal Broker refresh is required, or the Universal Broker started task must be restarted.

Configuration Options

This section describes the configuration options used to execute Universal Control Manager for z/OS.

Configuration Options Categories

Table 4.2, below, categorizes the configuration options into logical areas of application.

Category	Description
Command	Control command to execute.
Remote	Network address of the remote system.
User	User account the Control command executes with on the remote system.
Certificates	X.509 certificate related options.
Events	Options used to define event generation.
Local	Options required for local broker registration.
Messages	Universal Control message options.
Network	Options used to control the process of network data.
Options	Alternative methods to specify command options.
Miscellaneous	Options use to display command help and program versions.

Table 4.2 Universal Control Manager for z/OS - Configuration Options Categories

The UCTL Manager configuration options for each category are summarized in the following tables. Each **Option Name** is a link to detailed information about that option in the Universal Control 3.2.0 Reference Guide.

Certificate Category Options

Option Name	Description
CA_CERTIFICATES	ddname of the PEM-formatted trusted CA X.509 certificates
CERTIFICATE	ddname of Manager's PEM-formatted X.509 certificate.
CERTIFICATE_REVOCATION_LIST	Location of Manager's PEM-formatted CRL.
PRIVATE_KEY	ddname of Manager's PEM-formatted RSA private key.
PRIVATE_KEY_PWD	Password for the Manager's PRIVATE_KEY.
SSL_IMPLEMENTATION	Secure Socket Layer (SSL) implementation to be used for network communications
VERIFY_HOST_NAME	Specification that the Broker's X.509 certificate host name field must be verified.
VERIFY_SERIAL_NUMBER	Specification that the Broker's X.509 certificate serial number field must be verified.

Command Category Options

Option Name	Description
COMMAND_ID	Identity of the started component.
REFRESH_CMD	Instruction to a Broker to refresh configuration data.
START_CMD	Instruction to a Broker to start a component.
STOP_CMD	Instruction to stop a component being executed by a Broker.

Events Category Options

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
EVENT_GENERATION	Events to be generated as persistent events.

Local Category Options

Option Name	Description
SYSTEM_ID	Local Universal Broker with which the Universal Control Manager must register

Messages Category Options

Option Name	Description
MESSAGE_LANGUAGE	Language of messages written.
MESSAGE_LEVEL	Level of messages written.

Miscellaneous Category Options

Option Name	Description
HELP	Write command option help.
VERSION	Write program version.

Network Category Options

Option Name	Description
CODE_PAGE	Code page used to translate text data to and from the network.
CTL_SSL_CIPHER_LIST	SSL cipher list for the control session.
NETWORK_DELAY	Maximum number of seconds considered acceptable to wait for data communications.

Options Category Options

Option Name	Description
COMMAND_FILE_ENCRYPTED	Encrypted command file.
COMMAND_FILE_PLAIN	Plain text command file.
ENCRYPTION_KEY	Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED.

Remote Category Options

Option Name	Description
HOSTNAME_RETRY_COUNT	Number of host connection attempts before ending with an error.
OUTBOUND_IP	Host or IP address to use for all outgoing IP connections.
REMOTE_HOST	TCP/IP host name of the remote Broker.
REMOTE_PORT	TCP/IP port number of the remote Broker.

User Category Options

Option Name	Description
USER_ID	User ID or account with which to execute the Control command.
USER_PASSWORD	Password associated with USER_ID.

Command Line Syntax

Figure 4.3, below, illustrates the command line syntax – using the command line, long form of the configuration options – of Universal Control Manager for z/OS.

```
uctl
{ -start compname [-cmdid id] | -stop compID [-userid user [-pwd password] ] |
      -refresh [compname] }
-host hostaddress
[-file ddname | -encryptedfile ddname [-key key] ]  *
[-port port]
[-cmdid id]
[-hostname_retry_count count]
[-outboundip host]
[-ssl_implementation {openssl | system}
[-system_id ID]
[-lang language]
[-level {trace|audit|info|warn|error}]
[-ca_certs ddname [-verify_host_name {yes|no|hostname}]
   [-verify_serial_number number] ]
[-cert ddname -private_key ddname [-private_key_pwd password] ]
[-crl ddname]
[-codepage codepage]
[-ctl_ssl_cipher_list cipherlist]
[-delay seconds]

uctl
{-help | -version}
```

* The command file (`-file` or `-encryptedfile`) can contain some or all required and/or optional configuration options, including a control request and `-host`. If a command file is specified on the command line, and it contains the required control request and `-host` options, those options do not have to be specified additionally on the command line.

Figure 4.3 Universal Control Manager for z/OS - Command Line Syntax

4.2.2 Examples of UCTL Manager for z/OS

This section contains examples demonstrating the use of Universal Control Manager for z/OS. Each example is found in the Universal Control SUNVSAMP library.

The examples assume that Universal Control Server is installed on a remote system named **dallas**. The user ID and password used in the examples must be changed to a valid user ID and password for the remote system.

The following list provides a link to each example.

- Stop Component Example
- Start Component Example
- Refresh Component Example

Stop Component Example

This example stops a component on a remote system. The sample JCL is located in member UCTSAM1.

Figure 4.4, below, illustrates the JCL.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UCTLPRC
//SYSIN    DD  *
 -stop 999234133 -host dallas -userid joe -pwd akkSdiq
/*
```

Figure 4.4 Universal Control for z/OS - Stop Example

The JCL procedure **UCTLPRC** is used to execute the stop request.

The stop request is sent to a remote system named **dallas** for execution.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-stop	Specifies the component to stop.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Specifies the remote user ID with which to execute the stop request.
-pwd	Specifies the password for the user ID.

Start Component Example

This example starts a component on a remote system.

Figure 4.5, below, illustrates the command.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UCTLPRC
//SYSIN    DD  *
 -start uems –cmdid “UEM-dallas” –host dallas –userid joe -pwd akkSdiq
/*
```

Figure 4.5 Universal Control for z/OS - Start Component Example

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-start	Name of the component to start on the remote system.
-cmdid	Assigns a command identifier of **"UEM-dallas"** to the started component.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Remote user ID with which to execute the Universal Control Server process. The started component, in fact, will execute with the Universal Broker’s security context.
-pwd	Password for the user ID.

Refresh Component Example

This example refreshes a component on a remote system.

Figure 4.6, below, illustrates the command.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UCTLPRC
//SYSIN    DD  *
 -refresh uems -cmdid "UEM-dallas" -host dallas -userid joe -pwd akkSdiq
/*
```

Figure 4.6 Universal Control for z/OS - Refresh Component Example.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-refresh	Type of component to refresh on the remote system.
-cmdid	Assigns a command identifier of **"UEM-dallas"** to the started component.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Remote user ID with which to execute the Universal Control Server process.
-pwd	Password for the user ID.

4.2.3 Security

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

Universal Control security concerns are:

1. Access to Universal Control's files.
2. Privacy and integrity of transmitted network data.
3. RACF access to the remote system with a specific user identity.

Data Set Permissions

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

Eligible users of Universal Control require read access to:

- National language support library **SUNVNLS**
- Load library **SUNVLOAD**

RACF Protection

The Universal Control Manager 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 Universal Products 3.2.0 Installation Guide for complete details on installing and administering Universal Control Manager RACF profiles.

4.3 Universal Control Manager for Windows

This chapter provides information on Universal Control (UCTL) Manager specific to the Windows operating system.

UCTL Manager for Windows is a console application that can be run either from:

- Command prompt
- Universal Configuration Manager

Command Prompt

UCTL Manager runs as a command line program. It provides a command line interface to remote computers running the UCTL Server. On the command line, you must specify the control request that you want the UCTL Manager to execute.

Universal Configuration Manager

The Universal Configuration Manager provides a single interface from which active components can be listed and selected for termination. A list of active components can be obtained from any machine that is running Universal Broker.

See Section 2.4 Universal Configuration Manager for information on how to access and use Universal Configuration Manager.

4.3.1 Usage

This section describes the control requests, configuration and configuration options, and command line syntax of UCTL Manager for Windows.

Control Requests

UCTL Manager for Windows supports all three Universal Control control requests:

1. Start (START_CMD option)
2. Stop (STOP_CMD option)
3. Refresh (REFRESH_CMD option)

One of these control request options must be specified on the command line (or in a command file) for each execution of Universal Control Manager.

Section 4.3.2 Examples of UCTL Manager for Windows provides an example of each request.

Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCTL Manager.
- Setting options and preferences for a single execution of UCTL Manager.

Configuration options are read from the following sources:

1. Command line
2. Command file
3. Environment variables
4. Configuration file

The order of precedence is the same as the list above; command line being the highest, and configuration file being the lowest. That is, options specified via a command line override options specified via a command file, and so on. The UCTL Manager configuration file is provided to the manager by the local Universal Broker with which it registers.

The configuration file, **`uctl.conf`**, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Although configuration 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 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 (see Section 2.4 Universal Configuration Manager).

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCTL Manager.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Note: For any changes made directly to the UCTL Manager configuration file to become active, a Universal Broker refresh is required, or the Universal Broker service must be restarted. Changes made by the Universal Configuration Manager do not require any additional action for the options to become active.

Configuration Options

This section describes the configuration options used to execute Universal Control Manager for Windows.

Configuration Options Categories

Table 4.3, below, categorizes configuration options into logical areas of application.

Category	Description
Certificates	X.509 certificate related options.
Command	Control command to execute.
Events	Options used to define event generation.
Installation	Options that specify installation requirements, such as directory locations.
Messages	Universal Control message options.
Miscellaneous	Options use to display command help and program versions.
Network	Options used to control the process of network data.
Options	Alternative methods to specify command options.
Remote	Network address of the remote system.
User	User account the Control command executes with on the remote system.

Table 4.3 Universal Control Manager for Windows - Command Options Categories

The Universal Control Manager command options for each of the categories listed in Table 4.3 are summarized in the following tables. Each Option Name is a link to detailed information about that option in the Universal Control 3.2.0 Reference Guide.

Certificate Category Options

Option Name	Description
CA_CERTIFICATES	Location of the PEM-formatted trusted CA X.509 certificates.
CERTIFICATE	Location of Manager's PEM-formatted X.509 certificate.
CERTIFICATE_REVOCATION_LIST	Location of Manager's PEM-formatted CRL.
PRIVATE_KEY	Location of Manager's PEM-formatted RSA private key.
PRIVATE_KEY_PWD	Password for the Manager's PRIVATE_KEY.
VERIFY_HOST_NAME	Specification that the Universal Broker's X.509 certificate host name field must be verified.
VERIFY_SERIAL_NUMBER	Specification that the Universal Broker's X.509 certificate serial number field must be verified.

Command Category Options

Option Name	Description
COMMAND_ID	Identity of the started component.
REFRESH_CMD	Instruction to a Broker to refresh configuration data.
START_CMD	Instruction to a Universal Broker to start a component.
STOP_CMD	Instruction to stop a component being executed by a Broker.

Events Category Options

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
EVENT_GENERATION	Events to be generated as persistent events.

Installation Category Options

Option Name	Description
INSTALLATION_DIRECTORY	Directory in which Universal Control Server is installed.

Messages Category Options

Option Name	Description
MESSAGE_LANGUAGE	Language of messages written.
MESSAGE_LEVEL	Level of messages written.
NLS_DIRECTORY	Location of UMC and UTT files

Miscellaneous Category Options

Option Name	Description
HELP	Write command option help.
VERSION	Write program version.

Network Category Options

Option Name	Description
CODE_PAGE	Code page used to translate text data to and from the network.
CTL_SSL_CIPHER_LIST	SSL cipher list for the control session.
NETWORK_DELAY	Maximum number of seconds considered acceptable to wait for data communications.

Options Category Options Summary

Option Name	Description
COMMAND_FILE_ENCRYPTED	Encrypted command file.
COMMAND_FILE_PLAIN	Plain text command file.
ENCRYPTION_KEY	Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED.

Remote Category Options

Option Name	Description
HOSTNAME_RETRY_COUNT	Number of host connection attempts before ending with an error.
OUTBOUND_IP	Host or IP address to use for all outgoing IP connections.
REMOTE_HOST	TCP/IP host name of the remote Broker.
REMOTE_PORT	TCP/IP port number of the remote Broker.

User Category Options

Option Name	Description
USER_ID	User ID or account with which to execute the Control command.
USER_PASSWORD	Password associated with USER_ID.

Command Line Syntax

Figure 4.3, below, illustrates the command options syntax — using the command line, long form of the configuration options — of Universal Control Manager for Windows.

```
uctl
{ -start compname [-cmdid id] | -stop compID [-userid user [-pwd password] ] |
      -refresh [compname] }
-host hostaddress
[-file filename | -encryptedfile filename [-key key] ]  *
[-port port]
[-cmdid id]
[-hostname_retry_count count]
[-outboundip host]
[-lang language]
[-level {trace|audit|info|warn|error}]
[-ca_certs file [-verify_host_name {yes|no|hostname}]
   [-verify_serial_number number] ]
[-cert file -private_key file [-private_key_pwd password] ]
[-crl file]
[-codepage codepage]
[-ctl_ssl_cipher_list cipherlist]
[-delay seconds]

uctl
{-help | -version}
```

* The command file (`-file` or `-encryptedfile`) can contain some or all required and/or optional configuration options, including a control request and `-host`. If a command file is specified on the command line, and it contains the required control request and `-host` options, those options do not have to be specified additionally on the command line.

Figure 4.7 Universal Control Manager for Windows - Command Syntax

4.3.2 Examples of UCTL Manager for Windows

This section contains examples demonstrating the use of Universal Control Manager for Windows command line interface.

The following list provides a link to each example.

- Stop Component Example
- Start Component Example
- Refresh Component Example

Stop Component Example

This example stops a component on a remote system.

Figure 4.8, below, illustrates the command.

```
uctl –stop 10739132 -host dallas -userid joe –pwd akkSdiq
```

Figure 4.8 Universal Control for Windows - Stop Component Example

Command Line Options

The command line options used in this example are:

Option	Description
`-stop`	Component ID on the remote system to stop.
`-host`	Directs the command to a computer with a host name of **`dallas`**.
`-userid`	Remote user ID with which to execute the command. This must match the user ID originally used to start the component.
`-pwd`	Password for the user ID.

Start Component Example

This example starts a component on a remote system.

Figure 4.9, below, illustrates the command.

```
uctl –start uems –cmdid “UEM-dallas” -host dallas -userid joe –pwd akkSdiq
```

Figure 4.9 Universal Control for Windows - Start Component Example

Command Line Options

The command line options used in this example are:

Option	Description
-start	Name of the component to start on the remote system.
-cmdid	Assigns a command identifier of **"UEM-dallas"** to the started component.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Remote user ID with which to execute the Universal Control Server process. The started component, in fact, will execute with the Universal Broker’s security context.
-pwd	Password for the user ID.

Refresh Component Example

This example refreshes a component on a remote system.

Figure 4.10, below, illustrates the command.

```
uctl –refresh uems –cmdid “UEM-dallas” -host dallas -userid joe –pwd akkSdiq
```

Figure 4.10 Universal Control for Windows -Refresh Component Example.

Command Line Options

The command line options used in this example are:

Option	Description
`-refresh`	Type of component to refresh on the remote system.
`-cmdid`	Assigns a command identifier of **`"UEM-dallas"`** to the started component.
`-host`	Directs the command to a computer with a host name of **`dallas`**.
`-userid`	Remote user ID with which to execute the Universal Control Server process.
`-pwd`	Password for the user ID.

4.3.3 Security

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

Universal Control security concerns are:

- Access to Universal Control files and directories
- Access to Universal Control configuration options
- Privacy and integrity of transmitted network data

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` subdirectory of the Universal Products installation directory. If Universal Control Manager is installed on an `NTFS` partition, these file permissions are set automatically during the installation.

Universal Configuration Manager

The Universal Configuration Manager can be executed only by accounts in the Administrator group.

4.4 Universal Control Manager for UNIX

This chapter provides information on Universal Control (UCTL) Manager specific to the UNIX operating system.

UCTL Manager for UNIX runs as a command line program. It provides a command line interface to remote computers running the UCTL Server. On the command line, you must specify the control request that you want the UCTL Manager to execute.

4.4.1 Usage

This section describes the control requests, configuration and configuration options, and command line syntax of UCTL Manager for UNIX.

Control Requests

UCTL Manager for UNIX supports all three Universal Control control requests:

1. Start (START_CMD option)
2. Stop (STOP_CMD option)
3. Refresh (REFRESH_CMD option)

Section 4.4.2 Examples of UCTL Manager for UNIX provides an example of each request.

Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCTL Manager.
- Setting options and preferences for a single execution of UCTL Manager.

Configuration options are read from the following sources:

1. Command line
2. Command file
3. Environment variables
4. Configuration file

The order of precedence is the same as the list above; command line being the highest, and configuration file being the lowest. That is, options specified via a command line override options specified via a command file, and so on. The UCTL Manager configuration file is provided to the manager by the local Universal Broker with which it registers.

The configuration file, **`uctl.conf`**, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCTL Manager.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Note: For any changes to the UCTL Manager configuration file to become active, a Universal Broker refresh is required, or the Universal Broker daemon task must be restarted.

Configuration Options

This section describes the configuration options used to execute UCTL Manager for UNIX.

Configuration Options Categories

Table 4.4, below, categorizes configuration options into logical areas of application.

Category	Description
Certificates	X.509 certificate related options.
Command	Control command to execute.
Events	Options used to define event generation.
Installation	Options that specify installation requirements, such as directory locations.
Local	Options required for local broker registration.
Messages	Universal Control message options.
Miscellaneous	Options use to display command help and program versions.
Network	Options used to control the process of network data.
Options	Alternative methods to specify command options.
Remote	Network address of the remote system.
User	User account the Control command executes with on the remote system.

Table 4.4 Universal Control Manager for UNIX - Command Options Categories

The UCTL Manager configuration options for each category are summarized in the following tables. Each **Option Name** is a link to detailed information about that option in the Universal Control 3.2.0 Reference Guide.

Certificate Category Options

Option Name	Description
CA_CERTIFICATES	Location of the PEM-formatted trusted CA X.509 certificates.
CERTIFICATE	Location of Manager's PEM-formatted X.509 certificate.
CERTIFICATE_REVOCATION_LIST	Location of Manager's PEM-formatted CRL.
PRIVATE_KEY	Location of Manager's PEM-formatted RSA private key.
PRIVATE_KEY_PWD	Password for the Manager's PRIVATE_KEY.
VERIFY_HOST_NAME	Specification that the Universal Broker's X.509 certificate host name field must be verified.
VERIFY_SERIAL_NUMBER	Specification that the Universal Broker's X.509 certificate serial number field must be verified.

Command Category Options

Option Name	Description
COMMAND_ID	Identity of the started component.
START_CMD	Instruction for the Universal Broker to start a component.
STOP_CMD	Instruction to stop a component being executed by a Broker.
REFRESH_CMD	Instruction to a Broker to refresh configuration data.

Events Category Options

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
EVENT_GENERATION	Events to be generated as persistent events.

Installation Category Options

Option Name	Description
INSTALLATION_DIRECTORY	Directory in which Universal Control Server is installed.

Local Category Options

Option Name	Description
BIF_DIRECTORY	Broker Interface File (BIF) directory where the Universal Broker interface file is located.
PLF_DIRECTORY	Program Lock File (PLF) directory where the program lock files are located.

Messages Category Options

Option Name	Description
MESSAGE_LANGUAGE	Language of messages written.
MESSAGE_LEVEL	Level of messages written.
NLS_DIRECTORY	Location of UMC and UTT files.

Miscellaneous Category Options

Option Name	Description
HELP	Write command option help.
VERSION	Write program version.

Network Category Options

Option Name	Description
CODE_PAGE	Code page used to translate text data to and from the network.
CTL_SSL_CIPHER_LIST	SSL cipher list for the control session.
NETWORK_DELAY	Maximum number of seconds considered acceptable to wait for data communications.

Options Category Options

Option Name	Description
COMMAND_FILE_ENCRYPTED	Encrypted command file.
COMMAND_FILE_PLAIN	Plain text command file.
ENCRYPTION_KEY	Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED.

Remote Category Options

Option Name	Description
HOSTNAME_RETRY_COUNT	Number of host connection attempts before ending with an error.
OUTBOUND_IP	Host or IP address to use for all outgoing IP connections.
REMOTE_HOST	TCP/IP host name of the remote Broker.
REMOTE_PORT	TCP/IP port number of the remote Broker.

User Category Options

Option Name	Description
USER_ID	User ID or account with which to execute the Control command.
USER_PASSWORD	Password associated with USER_ID.

Command Line Syntax

Figure 4.3, below, illustrates the command line syntax – using the command line, long form of the configuration options – of Universal Control Manager for UNIX.

```
uctl
{ -start compname [-cmdid id] | -stop compID [-userid user [-pwd password] ] |
      -refresh [compname] }
-host hostaddress
[-file filename | -encryptedfile filename [-key key] ]  *
[-port port]
[-cmdid id]
[-hostname_retry_count count]
[-outboundip host]
[-bif_directory directory]
[-plf_directory directory]
[-lang language]
[-level {trace|audit|info|warn|error}]
[-ca_certs file [-verify_host_name {yes|no|hostname}]
   [-verify_serial_number number] ]
[-cert file -private_key file [-private_key_pwd password] ]
[-crl file]
[-codepage codepage]
[-ctl_ssl_cipher_list cipherlist]
[-delay seconds]

uctl
{-help | -version}
```

* The command file (`-file` or `-encryptedfile`) can contain some or all required and/or optional configuration options, including a control request and `-host`. If a command file is specified on the command line, and it contains the required control request and `-host` options, those options do not have to be specified additionally on the command line.

Figure 4.11 Universal Control Manager for UNIX - Command Line Syntax

4.4.2 Examples of UCTL Manager for UNIX

This section contains examples demonstrating the use of Universal Control Manager for UNIX.

The following list provides a link to each example.

- Stop Component Example
- Start Component Example
- Refresh Component Example

Stop Component Example

This example stops a component on a remote system.

Figure 4.12, below, illustrates the command.

```
uctl –stop 10739132 -host dallas -userid joe –pwd akkSdiq
```

Figure 4.12 Universal Control Manager for UNIX - Stop Component Example 1

Command Line Options

The command line options used in this example are:

Option	Description
`-stop`	Name of the component to stop.
`-host`	Directs the command to a computer with a host name of **`dallas`**.
`-userid`	Remote user ID with which to execute the command. This must match the user ID originally used to start the component.
`-pwd`	Password for the user ID.

Start Component Example

This example starts a component on a remote system.

Figure 4.13, below, illustrates the command.

```
uctl –start uems –cmdid “UEM-dallas” -host dallas -userid joe –pwd akkSdiq
```

Figure 4.13 Start Component Example.

Command Line Options

The command line options used in this example are:

Option	Description
-start	Name of the component to start on the remote system.
-cmdid	Assigns a command identifier of **"UEM-dallas"** to the started component.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Remote user ID with which to execute the Universal Control Server process. The started component, in fact, will execute with the Universal Broker’s security context.
-pwd	Password for the user ID.

Refresh Component Example

This example refreshes a component on a remote system.

Figure 4.14, below, illustrates the command.

```
uctl –refresh uems –cmdid “UEM-dallas” -host dallas -userid joe –pwd akkSdiq
```

Figure 4.14 Refresh Component Example.

Command Line Options

The command line options used in this example are:

Option	Description
`-refresh`	Type of component to refresh on the remote system.
`-cmdid`	Assigns a command identifier of **`"UEM-dallas"`** to the started component.
`-host`	Directs the command to a computer with a host name of **`dallas`**.
`-userid`	Remote user ID with which to execute the Universal Control Server process.
`-pwd`	Password for the user ID.

4.4.3 Security

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

Universal Control security concerns are:

1. Access to Universal Control files and directories
2. Access to Universal Control configuration options
3. Privacy and integrity of transmitted network data

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.

This most likely means only an administration group should have write access. Eligible users of Universal Control require read access to the message catalogs (`*.umc` files) in the NLS directory.

Configuration Files

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

4.5 Universal Control Manager for OS/400

This chapter provides information on Universal Control (UCTL) Manager specific to the OS/400 operating system.

Universal Control Manager for OS/400 runs via a command interface. It provides a command line interface to remote computers running the UCTL Server. On the command line, you must specify the control request that you want the UCTL Manager to execute.

4.5.1 Usage

This section describes the command execution environments, control requests, configuration and configuration options, and command line syntax of UCTL Manager for OS/400.

Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This chapter references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Control, substitute the tagged names for the untagged names in these references.

Command Execution Environment

The command is valid in all environments:

- Batch input streams
- CL programs
- REXX procedures
- CL ILE modules
- Interactive processing
- Passed to the system program QCMDEXC (or QCAEXEC) for processing

Control Requests

UCTL Manager for OS/400 supports all three Universal Control control requests:

1. Start (START_CMD option)
2. Stop (STOP_CMD option)
3. Refresh (REFRESH_CMD option)

Section 4.5.2 Examples of UCTL Manager for OS/400 provides an example of each request.

Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCTL Manager.
- Setting options and preferences for a single execution of UCTL Manager.

UCTL Manager for OS/400 configuration options are read from the following sources:

1. STRUCT parameters
2. Environment variables
3. Configuration file

The order of precedence is the same as the list above; STRUCT parameters being the highest, and configuration file being the lowest. That is, options specified via STRUCT parameters override options specified via environment variables, and so on.

The configuration file, **`UNVPRD320/UNVCONF(UCTL)`**, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCTL Manager.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Configuration Options

This section describes the configuration options used to execute Universal Control Manager for OS/400.

Configuration Options Categories

Table 4.5, below, categorizes configuration options into logical areas of application.

Category	Description
Certificates	X.509 certificate related options.
Command	Control command to execute.
Events	Options used to define event generation.
Local	Options required for local broker registration.
Messages	Universal Control message options.
Miscellaneous	Options use to display command help and program versions.
Network	Options used to control the process of network data.
Options	Alternative methods to specify command options.
Remote	Network address of the remote system.
User	User account that the Control command executes with on the remote system.

Table 4.5 Universal Control Manager for OS/400 - Command Options Categories

The UCTL Manager configuration options for each category are summarized in the following tables. Each **Option Name** is a link to detailed information about that option in the Universal Control 3.2.0 Reference Guide.

Certificate Category Options

Option Name	Description
CA_CERTIFICATES	Location of the PEM-formatted trusted CA X.509 certificates.
CERTIFICATE	Location of Manager's PEM-formatted X.509 certificate.
CERTIFICATE_REVOCATION_LIST	Location of Manager's PEM-formatted CRL.
PRIVATE_KEY	Location of Manager's PEM-formatted RSA private key.
PRIVATE_KEY_PWD	The password for the Manager's PRIVATE_KEY.
VERIFY_HOST_NAME	Specification that the Universal Broker's X.509 certificate host name field must be verified.
VERIFY_SERIAL_NUMBER	Specification that the Universal Broker's X.509 certificate serial number field must be verified.

Command Category Options

Option Name	Description
COMMAND_ID	Identity of the started component.
REFRESH_CMD	Instruction to a Broker to refresh configuration data.
START_CMD	Instruction to the Universal Broker to start a component.
STOP_CMD	Instruction to stop a component being executed by a Broker.
REMOTE_HOST	Instruction to a Broker or component to refresh its configuration.

Events Category Options

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
EVENT_GENERATION	Events to be generated as persistent events.

Local Category Options

Option Name	Description
PLF_DIRECTORY	Program Lock File (PLF) directory where the program lock files are located.

Messages Category Options

Option Name	Description
MESSAGE_LANGUAGE	Language of messages written.
MESSAGE_LEVEL	Level of messages written.

Miscellaneous Category Options

Option Name	Description
VERSION	Write program version.

Network Category Options

Option Name	Description
CODE_PAGE	Code page used to translate text data to and from the network.
CTL_SSL_CIPHER_LIST	SSL cipher list for the control session.
NETWORK_DELAY	Maximum number of seconds considered acceptable to wait for data communications.

Options Category Options

Option Name	Description
COMMAND_FILE_ENCRYPTED	Encrypted command file.
COMMAND_FILE_PLAIN	Plain text command file.
ENCRYPTION_KEY	Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED.

Remote Category Options

Option Name	Description
HOSTNAME_RETRY_COUNT	Number of host connection attempts before ending with an error.
OUTBOUND_IP	Host or IP address to use for all outgoing IP connections.
REMOTE_HOST	TCP/IP host name of the remote Broker.
REMOTE_PORT	TCP/IP port number of the remote Broker.

User Category Options

Option Name	Description
USER_ID	User ID or account with which to execute the Control command.
USER_PASSWORD	Password associated with USER_ID.

Command Line Syntax

The syntax shows the CL Command parameter followed by UNIX/CALL options (in parentheses). These options would be used to invoke Universal Command Manager on a different platform. They are provided to help the user associate STRUCT command options with UCTL command line options on other platforms.

Figure 4.15, below, illustrates the command line syntax — using the STRUCT parameter form of the configuration options — of Universal Control Manager for Windows.

```
STRUCT
{ START(compname) [CMDID(id)] | STOP(compID) [USER(user) [PWD(password)] ] |
      REFRESH ({yes|no}) [RFSHCMPNM(compname)] }
HOST(hostaddress)
[CMDFILE(filename) [CMDMBR(member)] | ECMFILE(filename) [ECMMBR(member)]
      [KEY(key)] ]
[PORT(port)]
[CMDID(id)]
[HSTNMRTYCT(count)]
[OUTBOUNDIP(host)]
[MSGLANG(language)]
[MSGLEVEL(*{trace|audit|info|warn|error})]
[CACERTS(file [lib]) [VFYHSTNM({yes|no|hostname})] [VFYSERNUM(number)] ]
[CERT(file [lib]) PVTKEYF(file [lib]) [PVTKEYPWD(password)] ]
[CRLFILE(file [lib]) [CRLMBR(member)] ]
[CODEPAGE(codepage)]
[CTLCPHRLST(cipherlist)]
[DELAY(seconds)]
[PLFDIR(directory)]

STRUCT
VERSION(*{yes|no})
```

* The command file (CMDFILE or ECMFILE) can contain some or all required and/or optional configuration options, including a control request and HOST. If a command file is specified on the command line, and it contains the required control request and HOST options, those options do not have to be specified additionally on the command line.

Figure 4.15 Universal Control Manager for OS/400 - Command Options Syntax

4.5.2 Examples of UCTL Manager for OS/400

This section contains examples demonstrating the use of Universal Control Manager for OS/400.

The following list provides a link to each example.

- Stop Component Example
- Start Component Example
- Refresh Component Example

These examples reference the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Control, substitute the tagged names for the untagged names. (See Universal Products for OS/400 Commands in Section 4.3.1 Usage for further information.)

Stop Component Example

This example stops a component on a remote system.

Table 4.16, below, illustrates the command.

```
STRUCT STOP(10739132) HOST(dallas) USERID(joe) PWD(akkSdiq)
```

Figure 4.16 Universal Control for OS/400 - Stop Component Example

Command Line Options

The command line options used in this example are:

Option	Description
STOP	Component on the remote system to stop.
HOST	Directs the command to a computer with a host name of **dallas**.
USERID	Remote user ID with which to execute the stop request. This must match the user ID originally used to start the component.
PWD	Password for the user ID.

Start Component Example

This example starts a component on a remote system.

Figure 4.17, below, illustrates the command.

```
STRUCT START(uems) CMDID('UEM-dallas') HOST(dallas) USERID(joe) PWD(akkSdiq)
```

Figure 4.17 Start Component Example.

Command Line Options

The command line options used in this example are:

Option	Description
START	Component to start on the remote system.
CMDID	Assigns a command identifier of '**UEM-dallas**' to the started component
HOST	Directs the command to a computer with a host name of **dallas**.
USERID	Remote user ID with which to execute the Universal Control Server process. The started component, in fact, will execute with the Universal Broker's security context.
PWD	Password for the user ID.

Refresh Component Example

This example refreshes a component on a remote system.

Figure 4.18, below, illustrates the command.

```
STRUCT REFRESH(uems) CMDID('UEM-dallas') HOST(dallas) USERID(joe) PWD(akkSdiq)
```

Figure 4.18 Universal Control Manager for OS/400 - Refresh Component Example

Command Line Options

The command line options used in this example are:

Option	Description
REFRESH	Type of component to refresh on the remote system.
CMDID	Assigns a command identifier of '**UEM-dallas**' to the started component
HOST	Directs the command to a computer with a host name of **dallas**.
USERID	Remote user ID with which to execute the Universal Control Server process.
PWD	Password for the user ID.

4.5.3 Security

Universal Control Manager for OS/400 is designed to be a secure system. As the level of security rises, so does the administrative complexity of the system. Universal Control Manager has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

Universal Control security concerns are:

- Access to Universal Control files and directories
- Access to Universal Control configuration options
- The privacy and integrity of transmitted network data

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.

This most likely means only an administration group should have write access. Eligible users of Universal Control require read access to the message catalogs (`*.umc` files) in the NLS directory.

Configuration Files

Only trusted user accounts should have write access to Universal Control's configuration file.

4.6 Universal Control Manager for HP NonStop

This chapter provides information on Universal Control (UCTL) Manager specific to the UNIX operating system.

**Currently, HP NonStop runs Universal Control 2.1.1.
This chapter provides information for that version.**

Universal Control Manager for HP NonStop runs as a command line program. It provides a command line interface to remote computers running the UCTL Server. On the command line, you must specify a control request that you want UCTL Manager to execute.

4.6.1 Usage

This section describes the control requests, configuration and configuration options, and command line syntax of UCTL Manager for HP NonStop.

Control Requests

UCTL Manager for HP NonStop supports only the following two Universal Control control requests:

4. Stop (STOP_CMD option)
5. Refresh (REFRESH_CMD option)

Section 4.6.2 Examples of UCTL Manager for HP NonStop provides an example of each request.

Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCTL Manager.
- Setting options and preferences for a single execution of UCTL Manager.

Configuration options are read from the following sources:

1. Command line
2. Command file
3. Environment variables
4. Configuration file

The order of precedence is the same as the list above; command line being the highest, and configuration file being the lowest. That is, options specified via a command line override options specified via a command file, and so on.

The configuration file, **UCTLCFG**, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCTL Manager.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Configuration Options

This section describes the configuration options used to execute UCTL Manager for HP NonStop.

Configuration Options Categories

Table 4.19, below, categorizes configuration options into logical areas of application.

Category	Description
Command	Control command to execute.
Remote	Network address of the remote system.
User	User account the Control command executes with on the remote system.
Messages	Universal Control message options.
Network	Options used to control the process of network data.
Options	Alternative methods to specify command options.
Miscellaneous	Options use to display command help and program versions.
Installation	Options that specify installation requirements, such as, directory locations.

Figure 4.19 Universal Control Manager for HP NonStop - Command Options Categories

The UCTL Manager configuration options for each category are summarized in the following tables.

Each **Option Name** is a link to detailed information about that option in the Universal Control 3.2.0 Reference Guide.

Command Category Options

Option Name	Description
STOP_CMD	Instruction to stop a component being executed by a Broker.
REFRESH_CMD	Instruction to a Broker to refresh configuration data.

Installation Category Options

Option Name	Description
INSTALLATION_DIRECTORY	Directory in which the product is installed.

Messages Category Options

Option Name	Description
MESSAGE_LANGUAGE	Language of messages written.
MESSAGE_LEVEL	Level of messages written.

Miscellaneous Category Options

Option Name	Description
HELP	Write command option help.
VERSION	Write program version.

Network Category Options

Option Name	Description
CODE_PAGE	Code page used to translate text data to and from the network.
NETWORK_DELAY	Maximum number of seconds considered acceptable to wait for data communications.

Options Category Options

Option Name	Description
COMMAND_FILE_ENCRYPTED	Encrypted command file.
COMMAND_FILE_PLAIN	Plain text command file.
ENCRYPTION_KEY	Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED.

Remote Category Options

Option Name	Description
REMOTE_HOST	TCP/IP host name of the remote Broker.
REMOTE_PORT	TCP/IP port number of the remote Broker.

User Category Options

Option Name	Description
USER_ID	User ID or account with which to execute the Control command.
USER_PASSWORD	Password associated with USER_ID.

Command Line Syntax

Figure 4.3, below, illustrates the command options syntax of Universal Control Manager for HP NonStop.

Section Configuration Options provides a description of the options.

```
uctl
{ -stop compID [-userid user [-pwd password] ] | -refresh [compname] }
-host hostaddress
[-file filename | -encryptedfile filename [-key key] ]  *
[-port port]
[-lang language]
[-level {trace|audit|info|warn|error}]
[-codepage codepage]
[-delay seconds]

uctl
{-help | -version}
```

Figure 4.20 Universal Control Manager for HP NonStop - Command Options Syntax

4.6.2 Examples of UCTL Manager for HP NonStop

This section contains examples demonstrating the use of Universal Control Manager for HP NonStop

Stop Component Example

This example stops a component on a remote system.

Figure 4.21, below, illustrates the command.

```
run uctl –stop 10739132 -host dallas -userid joe –pwd akkSdiq
```

Figure 4.21 Universal Control Manager for HP NonStop - Stop Component Example 1

Command Line Options

The command line options used in this example are:

Option	Description
`-stop`	ID of the component on the remote system to stop.
`-host`	Directs the command to a computer with a host name of **`dallas`**.
`-userid`	Remote user ID with which to execute the command. This must match the user ID originally used to start the component.
`-pwd`	Password for the user ID.

4.6.3 Security

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

Universal Control security concerns are:

1. Access to Universal Control files and subvolumes.
2. Access to Universal Control configuration options.
3. Privacy and integrity of transmitted network data.

File Permissions

Only trusted user accounts should have write permission to the Universal Control Server installation subvolume and all of the files within it.

This most likely means that only an administration group should have write access. Eligible users of Universal Control require read access to the message catalogs in the **$SYSTEM.UNVNLS** subvolume.

Configuration Files

Only trusted user accounts should have write access to Universal Control's configuration file.

4.7 Universal Control Server for z/OS

This chapter documents the Universal Control (UCTL) Server at a detailed level, specific to the z/OS operating system.

Environment

The UCTL Server runs as z/OS UNIX System Services (USS) background process started by the Universal Broker. The address space name is **UCTSRV**. Its user identifier is inherited from the Broker address space.

As with all components managed by the Universal Broker, UCTL Server inherits the message language from the Universal Broker. All messages generated by the Universal Control Server are sent to Universal Broker for processing.

User Identification

UCTL Server can operate with user security active or inactive, based on the USER_SECURITY configuration option.

- With user security active, the UCTL Server requires the UCTL Manager to supply a valid z/OS user ID and a password. The user profile must have a properly defined OMVS segment.
- With user security inactive, the UCTL Server does not require the UCTL Manager to supply a valid user ID. Essentially, any operation that the UCTL Server is capable of can be requested by any UCTL Manager.

4.7.1 Component Definition

All Universal Products components managed by Universal Broker have a component definition. The component definition is a text file of options containing component-specific information required by Universal Broker. (For details on how Universal Broker manages components, see the Universal Broker 3.2.0 User Guide.)

The syntax of a component definition file is the same as a configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

The UCTL Server for z/OS component definition is located in the component definition library **`UNVCOMP`** allocated to the Universal Broker ddname **`UNVCOMP`**. The UCTL Server component definition member is **`UTSCMP00`**.

Table 4.6, below, identifies all of the options that comprise the UCTL Server for z/OS component definition.

Each **Option Name** is a link to detailed information about that component definition option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
AUTOMATICALLY_START	Specification for whether or not UCTL Server starts automatically when Universal Broker is started.
COMPONENT_NAME	Name by which the clients know the UCTL Server,
CONFIGURATION_FILE	Name of the UCTL Server's configuration file,
RUNNING_MAXIMUM	Maximum number of UCTL Servers that can run simultaneously,
START_COMMAND	Member name of the UCTL Server program,
WORKING_DIRECTORY	Directory used as the working directory of the UCTL Server,

Table 4.6 UCTL Server for z/OS - Component Definition Options

4.7.2 Configuration

UCTL Server configuration consists of defining runtime and default values. This section describes the Server configuration options.

See Section 2.2 Configuration for details on Universal Products configuration methods.

Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation. This file can be edited manually with any text editor.

The UCTL Server configuration file name is specified in the Universal Control Server component definition. The default name is **`UTSCFG00`**. The name refers to a member in the PDS allocated to the Universal Broker ddname **`UNVCONF`**.

Note: For any changes to the UCTL Server configuration file to become active, a Universal Broker refresh is required, or the Universal Broker started task must be restarted.

Configuration Options Summary

Table 4.7, below, identifies all of the UCTL Server for z/OS configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
CODE_PAGE	Code page used for text translation.
EVENT_GENERATION	Events to be generated as persistent events.
MESSAGE_LEVEL	Level of messages printed.
TMP_DIRECTORY	HFS directory name used for temporary files.
USER_SECURITY	Specification for whether or not user authentication is active.

Table 4.7 UCTL Server for z/OS - Configuration Options

4.7.3 Security

UCTL Server security concerns are:

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

File Permissions

Only trusted user accounts should have write permission to the UCTL Server installation data sets. No general user access is required.

Universal Control Server User ID

UCTL Server requires read access to its installation data sets and its HFS working directory (defined in the component definition).

User Authentication

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

Universal Access Control List

The UCTL Server uses the Universal Access Control List (UACL) file as an extra layer of security. The UACL file contains entries for the Universal Control Server. These entries contain Access Control List (ACL) rules that permit or deny access to the UCTL Server.

See Section 2.6 Universal Access Control List for details on the Universal Access Control List feature.

UACL Entries

The syntax of a UACL entry file is the same as the UCTL configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.8 identifies all UCTL Server for z/OS UACL entries.

Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Control 3.2.0 Reference Guide.

UACL Entry Name	Description
UCTL_ACCESS	Allows or denies access to UCTL Server services,
UCTL_REQUEST	Allows or denies access to UCTL Server services based on client identification and request type,

Table 4.8 UCTL Server for z/OS - UACL Entries

UACL Examples

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
```

4.8 Universal Control Server for Windows

This chapter documents the Universal Control (UCTL) Server at a detailed level, specific to the Windows family of operating systems.

Environment

The UCTL Server runs as a background process. It does not interact with a console or desktop.

As with all components managed by the Universal Broker, UCTL Server inherits the message language from the Universal Broker. All messages generated by the UCTL Server are sent to Universal Broker for processing.

User Identification

UCTL Server can operate with user security active or inactive, based on the USER_SECURITY configuration option.

- With user security active, the UCTL Server requires the UCTL Manager to supply a valid user ID for the local system and a password.
- With user security inactive, the UCTL Server does not require the UCTL Manager to supply a valid user ID. Essentially, any operation that the UCTL Server is capable of can be requested by any UCTL Manager.

4.8.1 Component Definition

All Universal Products components managed by Universal Broker have a component definition. The component definition is a text file of options containing component-specific information required by Universal Broker. (For details on how Universal Broker manages components, see the Universal Broker 3.2.0 User Guide.)

The syntax of a component definition file is the same as a configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Although component definition files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application is the recommended way to edit component definitions for Windows (see Section 2.4 Universal Configuration Manager).

Note: The component definitions for all Universal Products are identified in the Component Definitions property page of the Universal Broker (see Figure 4.22, below).

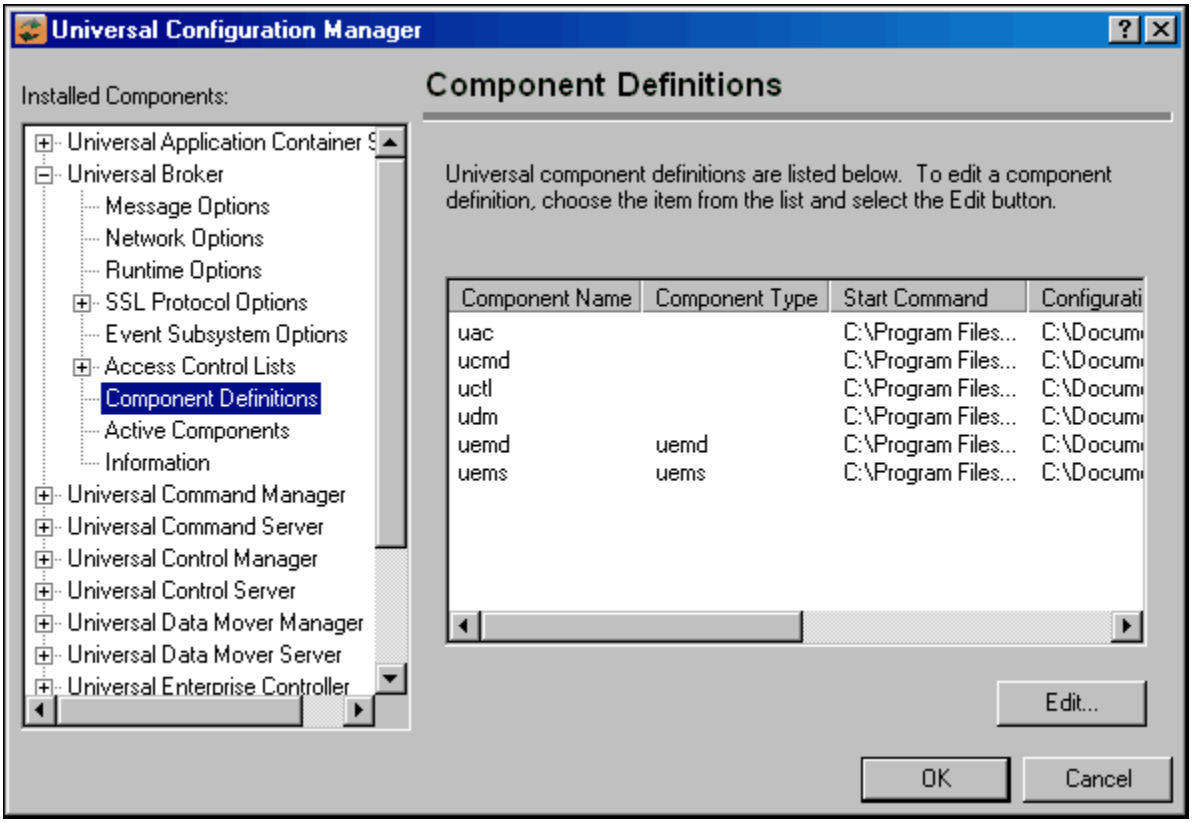

Figure 4.22 Universal Configuration Manager - Component Definitions

Table 4.9, below, identifies all of the options that comprise the UCTL Server for WIndows component definition.

Each **Option Name** is a link to detailed information about that component definition option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
AUTOMATICALLY_START	Specification for whether or not UCTL Server starts automatically when Universal Broker is started.
COMPONENT_NAME	Name by which the clients know the UCTL Server.
CONFIGURATION_FILE	Name of the UCTL Server configuration file.
RUNNING_MAXIMUM	Maximum number of UCTL Servers that can run simultaneously.
START_COMMAND	Full path name of the UCTL Server program.
WORKING_DIRECTORY	Directory used as the working directory of the UCTL Server.

Table 4.9 UCTL Server for Windows - Component Definition Options

4.8.2 Configuration

UCTL Server configuration consists of defining run-time and default values. This section describes the Server configuration options.

See Section 2.2 Configuration for details on Universal Products configuration methods.

Configuration File

The configuration file provides a simple method of specifying configuration values that will not change with each command invocation. This file can be edited manually with any text editor (for example, Notepad). See Section 2.2.6 Configuration File Syntax for details on configuration file syntax.

The UCTL Server configuration file name is specified in the Universal Control Server component definition.

However, the Universal Configuration Manager application, accessible via the Control Panel, is the recommended way to set Windows 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 (see Section 2.4 Universal Configuration Manager).

Note: For any changes made directly to the UCTL Server configuration file to become active, a Universal Broker refresh is required, or the Universal Broker service must be restarted. Changes made by the Universal Configuration Manager do not require any additional action for the options to become active.

Configuration Options Summary

Table 4.7, below, identifies all of the UCTL Server for Windows configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
CODE_PAGE	Code page used for text translation.
EVENT_GENERATION	Events to be generated as persistent events.
INSTALLATION_DIRECTORY	Base directory in which Universal Control Server is installed.
LOGON_METHOD	Method of how users are logged onto the system.
MESSAGE_LEVEL	Level of messages written.
NLS_ DIRECTORY	Location of UMC and UTT files.
TMP_DIRECTORY	Directory name used for temporary files.
USER_SECURITY	Specification for whether or not user authentication is active.

Table 4.10 UCTL Server for Windows - Configuration Options

4.8.3 Security

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

UCTL security concerns are:

- Access to Universal Control files and directories
- Universal Control user account
- Privacy and integrity of transmitted network data
- User authentication

File Permissions

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

This most likely means only the Administrators group should have write access. Eligible users of UCTL require read access to the message catalogs (`*.umc` files) in the `nls` subdirectory of the Universal Products 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.

Configuration Options

It is recommended that configuration options are set by the Universal Configuration Manager, which can only be executed by accounts in the Administrator group.

User Authentication

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

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 Section 4.7 LOGON_METHOD in the Universal Control 3.2.0 Reference Guide for more details.)

Universal Access Control List

The Universal Control Server uses the Universal Access Control List (UACL) file as an extra layer of security. The UACL file contains entries for the Universal Control Server. These entries contain Access Control List (ACL) rules that permit or deny access to the Universal Control Server.

See Section 2.6 Universal Access Control List for details on the Universal Access Control List feature.

UACL Entries

The syntax of a UACL file is the same as the Universal Control configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.8 identifies all UCTL Server for Windows UACL entries.

Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Control 3.2.0 Reference Guide.

UACL Entry Name	Description
UCTL_ACCESS	Allows or denies access to Universal Control Server services.
UCTL_REQUEST	Allows or denies access to Universal Control Server services based on client identification and request type.

Table 4.11 Universal Control for Windows - UACL Entries

Updating the Universal Control Server ACL Entries

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 (see Section 2.4 Universal Configuration Manager). From there, ACL entries can be added, changed, deleted or sorted (rules are applied in the order in which they are listed).

Figure 4.23, below, 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.

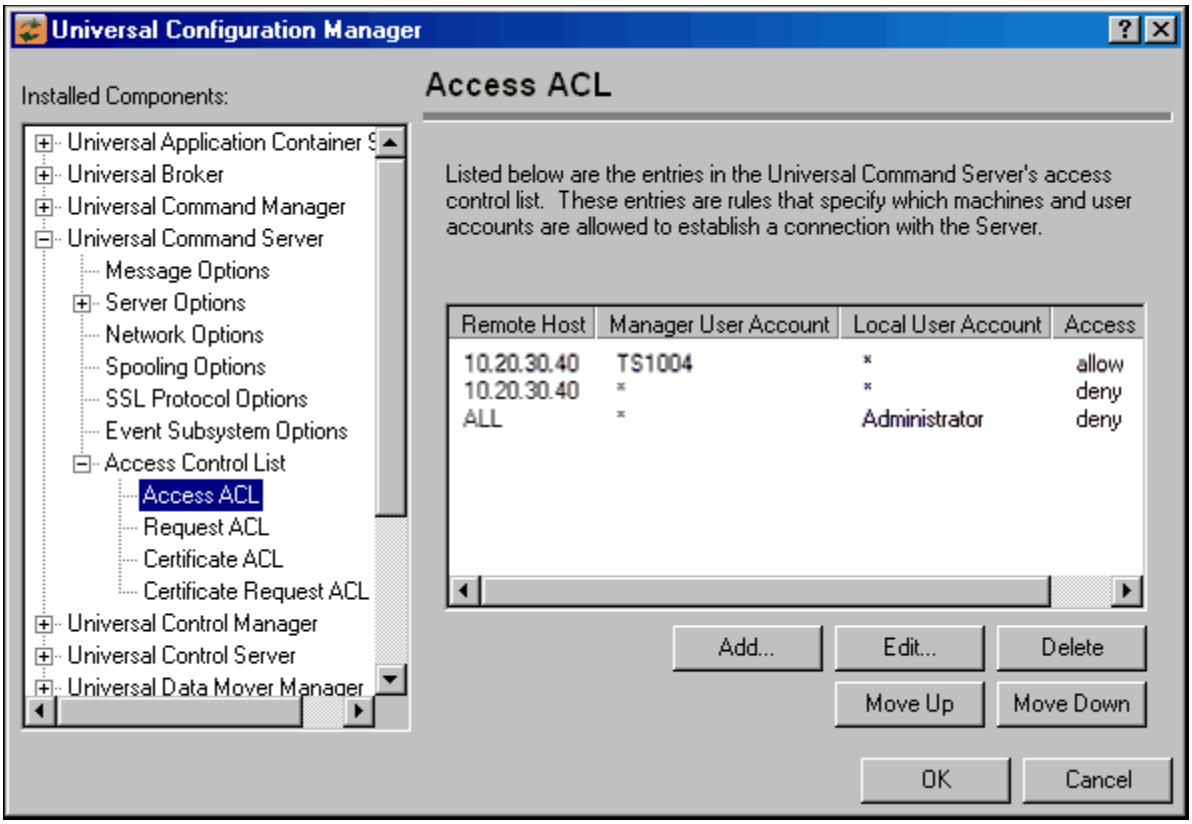

Figure 4.23 Universal Configuration Manager - Universal Control Server - Access ACL

4.9 Universal Control Server for UNIX

This chapter documents the Universal Control (UCTL) Server at a detailed level, specific to the UNIX operating system.

Environment

The Universal Control Server runs as a background process. It does not interact with a console.

As with all components managed by the Universal Broker, Universal Control Server inherits the message language from the Universal Broker. All messages generated by the Universal Control Server are sent to Universal Broker for processing.

User Identification

Universal Control Server can operate with user security active or inactive, based on the USER_SECURITY configuration option.

- With user security active, the Universal Control Server requires the Universal Control Manager to supply a valid user ID for the local system and a password.
- With user security inactive, the Universal Control Server does not require the Manager to supply a valid user ID. Essentially, any operation the Control Server is capable can be requested by any Control Manager.

4.9.1 Component Definition

All Universal Products components managed by Universal Broker have a component definition. The component definition is a text file of options containing component-specific information required by Universal Broker. (For details on how Universal Broker manages components, see the Universal Broker 3.2.0 User Guide.)

The syntax of a component definition file is the same as a configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.12, below, identifies all of the options that comprise the UCTL Server for UNIX component definition.

Each **Option Name** is a link to detailed information about that component definition option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
AUTOMATICALLY_START	Specification for whether or not UCTL Server starts automatically when Universal Broker is started.
COMPONENT_NAME	Name by which the clients know the UCTL Server.
CONFIGURATION_FILE	Name of the UCTL Server configuration file.
RUNNING_MAXIMUM	Maximum number of UCTL Servers that can run simultaneously.
START_COMMAND	Full path name of the UCTL Server program.
WORKING_DIRECTORY	Directory used as the working directory of the UCTL Server.

Table 4.12 UCTL Server for UNIX - Component Definition Options

4.9.2 Configuration

Universal Control Server configuration consists of defining runtime and default values. This section describes the Server configuration options.

See Section 2.2 Configuration for details on Universal Products configuration methods.

Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation. This file can be edited manually with any text editor.

The Universal Control Server configuration file name is specified in the Universal Control Server component definition. The default name is **`uctls.conf`**. Refer to the component definition file to determine the directory in which it is located.

See Section 2.2.6 Configuration File Syntax for details on configuration file syntax.

Note: For any changes made directly to the UCTL Server configuration file to become active, a Universal Broker refresh is required, or the Universal Broker service must be restarted.

Configuration Options Summary

Table 4.7, below, identifies all of the Universal Control Server for UNIX configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.
CODE_PAGE	Code page used for text translation.
EVENT_GENERATION	Events to be generated as persistent events.
INSTALLATION_DIRECTORY	Base directory in which Universal Control Server is installed.
MESSAGE_LEVEL	Level of messages written.
NLS_ DIRECTORY	Location of UMC and UTT files.
TMP_DIRECTORY	Directory name used for temporary files.
TRACE_DIRECTORY	Location of trace files.
USER_SECURITY	Specification for whether or not user authentication is active.

Table 4.13 UCTL Server for UNIX - Configuration Options

4.9.3 Security

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

Universal Control security concerns are:

1. Access to Universal Control files and directories
2. Universal Control user account
3. Privacy and integrity of transmitted network data
4. User authentication

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.

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.

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 UNIX operating system for verification; this is referred to as logging on the user.

Universal Access Control List

The Universal Control Server uses the Universal Access Control List (UACL) file as an extra layer of security. The UACL file contains entries for the Universal Control Server. These entries contain Access Control List (ACL) rules that permit or deny access to the Universal Control Server.

See Section 2.6 Universal Access Control List for details on the Universal Access Control List feature.

UACL Entries

The syntax of a UACL entry file is the same as the Universal Control configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.8 identifies all UCTL Server for UNIX UACL entries.

Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Control 3.2.0 Reference Guide.

UACL Entry Name	Description
UCTL_ACCESS	Allows or denies access to Universal Control Server services.
UCTL_REQUEST	Allows or denies access to Universal Control Server services based on client identification and request type.

Table 4.14 UCTL Server for UNIX - UACL Entries

UACL Examples

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
```

4.10 Universal Control Server for OS/400

This chapter documents the Universal Control Server at a detailed level, specific to the OS/400 operating system.

Environment

The Universal Control Server runs under the **UNVUBR320** subsystem's pre-start job **UNVSRV**. When the Broker receives a request for a Universal Command component, it passes the request to the **UCTSRV** program running under the **UNVSRV** pre-start job.

As with all components managed by the Universal Broker, Universal Control Server inherits the message language from the Universal Broker. All messages generated by the Universal Control Server are sent to Universal Broker for processing.

User Identification

Universal Control Server can operate with user security active or inactive, based on the USER_SECURITY configuration option.

- With user security active, the Server requires the Manager to supply a valid user ID and password for the local system. The user command executes with the user profile of the received user ID.
- With user security inactive, the Server does not require the Manager to supply a valid user ID. The user command executes with the user profile of the Server. The user profile of the Server is inherited from the Broker. The inherited profile is **UNVUBR320**; as installed, this profile provides a very high level of authority including *ALLOBJ.

Current Library and Working Directory

The current library and working directory of a user command depends on whether user security is active or inactive:

- With user security active, the user's current library is designated by the user profile and the working directory is the home directory of the user profile.
- With user security inactive, the current library is that for the user profile (the installation default, **UNVUBR320**) associated with the service program. Note that the default value used for the current library is **UNVTMP320**. Care should be taken to avoid name clashes and other consequences of multiple processes sharing a common current library and working directory.

4.10.1 Component Definition

All Universal Products components managed by Universal Broker have a component definition. The component definition is a text file of options containing component-specific information required by Universal Broker. (For details on how Universal Broker manages components, see the Universal Broker 3.2.0 User Guide.)

The default location for Universal Broker component definition files is **UNVPRD320/UNVCOMP**. The UCTL Server component member is **UCTL**.

The syntax of a component definition file is the same as a configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.12, below, identifies all of the options that comprise the UCTL Server for OS/400 component definition.

Each **Option Name** is a link to detailed information about that component definition option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
AUTOMATICALLY_START	Specification for whether or not UCTL Server starts automatically when Universal Broker is started.
COMPONENT_NAME	Name by which the clients know the UCTL Server.
CONFIGURATION_FILE	Name of the UCTL Server configuration file.
RUNNING_MAXIMUM	Maximum number of UCTL Servers that can run simultaneously.
START_COMMAND	Full path name of the UCTL Server program.
WORKING_DIRECTORY	Directory used as the working directory of the UCTL Server.

Table 4.15 UCTL Server for OS/400 - Component Definition Options

4.10.2 Configuration

Universal Control Server configuration consists of defining runtime and default values. This section describes the Server configuration options.

See Section 2.2 Configuration for details on Universal Products configuration methods.

Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation. This file can be edited manually with any text editor (for example, Notepad).

The Universal Control Server configuration file name is specified in the Universal Control Server component definition. The default file name is **`UNVPRD320/UNVCONF(UCTS)`**.

See Section 2.2.6 Configuration File Syntax for details on configuration file syntax.

Configuration Options Summary

Table 4.7, below, identifies all of the Universal Control Server for OS/400 configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
CODE_PAGE	Code page used for text translation.
MESSAGE_LEVEL	Level of messages written.
USER_SECURITY	Specification for whether or not user authentication is active.

Table 4.16 Universal Control Server for OS/400 - Configuration Options

4.10.3 Security

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

Universal Control security concerns are:

- Access to Universal Control files and directories
- Access to Universal Control configuration files
- Universal Control user account
- Privacy and integrity of transmitted network data
- User authentication

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.

Configuration Files

Only trusted user accounts should have management, existence, alter, add, update or delete authority to the Universal Control Server configuration files, and access to the libraries in which they reside.

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 Server authority is inherited from the Broker and the associated user profile (**`UNVUBR320`**) provides *ALLOBJ authority.

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 OS/400 operating system for verification; this is referred to as logging on the user.

Universal Access Control List

The Universal Control Server uses the Universal Access Control List (UACL) file as an extra layer of security. The UACL file contains entries for the Universal Control Server. These entries contain Access Control List (ACL) rules that permit or deny access to the Universal Control Server.

See Section 2.6 Universal Access Control List for details on the Universal Access Control List feature.

UACL Entries

The syntax of a UACL entry file is the same as the Universal Control configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.8 identifies all UCTL Server for OS/400 UACL entries.

Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Control 3.2.0 Reference Guide.

UACL Entry Name	Description
UCTL_ACCESS	Allows or denies access to Universal Control Server services.
UCTL_REQUEST	Allows or denies access to Universal Control Server services based on client identification and request type.

Table 4.17 Universal Control for OS/400 - UACL Entries

UACL Examples

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
uctl_cert_access   *,root,deny,auth
```

4.11 Universal Control Server for HP NonStop

This chapter documents the Universal Control (UCTL) Server at a detailed level, specific to the HP NonStop operating system.

**Currently, HP NonStop runs Universal Control 2.1.1.
This chapter provides information for that version.**

Environment

UCTL Server runs as a background OSS process. It does not interact with a console.

As with all components managed by the Universal Broker, the UCTL Server inherits the message language from the Universal Broker. All messages generated by the UCTL Server are sent to Universal Broker for processing.

User Identification

UCTL Server can operate with user security active or inactive, based on the USER_SECURITY configuration option.

- With user security active, the UCTL Server requires the UCTL Manager to supply a valid user ID for the local system and a password.
- With user security inactive, the UCTL Server does not require the UCTL Manager to supply a valid user ID. Essentially, any operation for which the UCTL Server is capable can be requested by any UCTL Manager.

4.11.1 Component Definition

All Universal Products components managed by Universal Broker have a component definition. The component definition is a text file of options containing component-specific information required by Universal Broker. (For details on how Universal Broker manages components, see the Universal Broker 3.2.0 User Guide.)

The syntax of a component definition file is the same as a configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.12, below, identifies all of the options that comprise the UCTL Server for HP NonStop component definition.

Each **Option Name** is a link to detailed information about that component definition option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
AUTOMATICALLY_START	Specification for whether or not UCTL Server starts automatically when Universal Broker is started.
COMPONENT_NAME	Name by which the clients know the UCTL Server.
CONFIGURATION_FILE	Name of the UCTL Server configuration file.
RUNNING_MAXIMUM	Maximum number of UCTL Servers that can run simultaneously.
START_COMMAND	Full path name of the UCTL Server program.
WORKING_DIRECTORY	Directory used as the working directory of the UCTL Server.

Table 4.18 UCTL Server for HP NonStop - Component Definition Options

4.11.2 Configuration

Universal Control Server configuration consists of defining runtime and default values. This section describes the Server configuration options.

See Section 2.2 Configuration for details on Universal Products configuration methods.

Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation. This file can be edited manually using the TACL EDIT command.

The Universal Control Server configuration file name is specified in the Universal Control Server component definition. The default name is **UCTLSCFG**. Refer to the component definition file to determine the subvolume in which it is located.

Configuration Options

Table 4.7, below, identifies all of the Universal Control Server for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Control 3.2.0 Reference Guide.

Option Name	Description
CODE_PAGE	Code page used for text translation.
MESSAGE_LEVEL	Level of messages written.
USER_SECURITY	Specification for whether or not user authentication is active.

Table 4.19 Universal Control Server for OS/400 - Configuration Options

See the Universal Control 3.2.0 Reference Guide for complete details on these options.

4.11.3 Security

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

Universal Control security concerns are:

1. Access to Universal Control files and subvolumes
2. Access to Universal Control configuration files
3. Universal Control user account
4. Privacy and integrity of transmitted network data
5. User authentication

File Permissions

Only trusted user accounts should have write permission to the Universal Control Server installation subvolume and all of the files within that subvolume.

Configuration Files

Only trusted user accounts should have write permission to the Universal Control Server configuration files, as well as add and delete access to the subvolume in which they reside.

Universal Control Server User ID

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

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 HP NonStop operating system for verification; this is referred to as logging on the user.

Universal Access Control List

The UCTL Server uses the Universal Access Control List (UACL) file as an extra layer of security. The UACL file contains entries for the UCTL Server. These entries contain Access Control List (ACL) rules that permit or deny access to the UCTL Server.

See Section 2.6 Universal Access Control List for details on the Universal Access Control List feature.

UACL Entries

The syntax of a UACL entry file is the same as the Universal Control configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 4.8 identifies all UCTL Server for HP NonStop UACL entries.

Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Control 3.2.0 Reference Guide.

UACL Entry Name	Description
UCTL_ACCESS	Allows or denies access to Universal Control Server services.

Table 4.20 UCTL Server for HP NonStop - UACL Entries

UACL Examples

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

```
uctl_access     10.20.30.,*,*,allow,auth
uctl_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.

```
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
```

Chapter 5
Universal Copy

5.1 Overview

Universal Copy provides a means to copy files from either:

- Manager to a Server
- Server to Manager

5.1.1 Usage

Universal Copy copies files specified on its command line to stdout or a specified output file. The files are concatenated in the order specified on the command line. If no files are specified, it copies from stdin.

The default transfer mode used for the Universal Copy command is binary. In order to force end-of-line character interpretation, mode of text must be specified as a parameter of the Universal Copy command.

The default mode of transfer for standard in, standard out, and standard error is **`text`**. If binary is required, mode of **`binary`** must be specified on the standard file parameters.

5.2 Universal Copy for Windows and UNIX

This section describes the configuration options and command line syntax of Universal Copy for the Windows and UNIX operating systems.

5.2.1 Configuration Options

Table 5.1 identifies all Universal Copy for Windows and UNIX configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
HELP	Write a description of the command options and their format.
MESSAGE_LEVEL	Level of messages that will be displayed.
MODE	Mode in which input files are read and output files are written.
OUTPUT	File name that data is written to instead of standard output.
REPLACE	Specification for whether or not the file name specified with the OUTPUT option is replaced if it already exists.
TRANSACTIONAL	Specification for whether or not the copy operation is performed in transactional mode.
VERSION	Writes the program version and copyright information.

Table 5.1 Universal Copy for Windows and UNIX - Configuration Options

5.2.2 Command Line Syntax

Figure 5.1, below, illustrates the syntax — using the long form of command line options — of Universal Copy for Windows and UNIX.

```
ucopy
[-level {trace|audit|info|warn|error}]
[-mode {binary|text}]
[-output filename [-transactional {yes|no}] [-replace {yes|no}] ]
[file ...]

ucopy
{ -version | -help }
```

Figure 5.1 Universal Copy for Windows and UNIX - Command Line Syntax

5.2.3 Command Operands

FILE

The file operand specifies the input files. Full or relative paths can be specified.

If no input files are specified, standard input is used.

z/OS USS

z/OS USS permits the specification of files located in the hierarchical file system (HFS) and z/OS data sets.

HFS files are specified simply as UNIX file names.

z/OS data sets are specified using the IBM USS **//** convention, which prefixes the data set name with the characters **//**. The syntax is as follows:

//['']data.set.name[(member)][']

In order for the USS shell to interpret the forward slash (/) characters correctly, the complete file name must be enclosed in double (**"**) quotation marks on the USS command line.

The data set name adheres to TSO naming conventions; if it is not enclosed in apostrophes, your USS user name is used as the high-level qualifier. For example, **"//my.data"** refers to data set **USERID.MY.DATA**.

5.3 Universal Copy for OS/400

This section describes the configuration options and command line syntax of Universal Copy for the OS/400 operating systems.

Note: Universal Copy became available for the OS/400 environment with PTF 0UC0104 (level 1.2.1).

5.3.1 Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This chapter references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Copy, substitute the tagged names for the untagged names in these references.

5.3.2 Description

The Universal Copy for OS/400 command is **STRUCP**.

STRUCP copies files specified by FRMFILE and FRMFILES parameters to STDOUT or to a file specified by the TOFILE parameter. The files are concatenated in the order specified, starting with FRMFILE and continuing with the FRMFILES list. If no files are specified, it copies from STDIN to STDOUT.

5.3.3 Configuration Options

Table 5.2, below, identifies all Universal Copy for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CPY_MODE	Copy mode for reading and writing files.
FRMFILE	Name of a file to copy.
FRMFILES	Names of additional files to copy.
FRMMBR	Name of a member in the file specified by FRMFILE.
INPUTMODE	Mode for opening the file for input.
MESSAGE_LEVEL	Level of messages displayed by Universal Copy.
OUTPUTMODE	Mode for opening the file for output.
REPLACE	Specification for whether or not existing output file should be replaced.
TOFILE	Name of the output file that receives the specified concatenated input files.
TOMBR	Name of a member within the file specified by TOFILE.
TRANSACTIONAL	Specification for whether or not the copy operation is performed in transactional mode.
VERSION	Writes the program version and copyright information.

Table 5.2 Universal Copy Configuration Options - OS/400

5.3.4 Command Line Syntax

Figure 5.2, below, illustrates the syntax — using the STRUCP parameter form of command line options — of the Universal Copy for OS/400.

```
STRUCP
[FRMFILE([{*lib|*curlib|library name}/] {*stdin|filename})
   [FRMMBR({*first|*all|member name})]
[FRMFILES( ([{*lib|*curlib|library name}/] file [*first|*all}])...)] ]
[TOFILE( [{*lib|*curlib|library name}/] {*stdout|filename})
   [TOMBR(member)] ]
[CPYMODE(*binary|*text|*savf)]

** Additional Options **
[REPLACE(*yes|*no)]
[INPUTMODE('option')]     Note: Overrides CPYMODE for input files.
[OUTPUTMODE('option')]    Note: Overrides CPYMODE for output files.
[MSGLEVEL(*trace|*audit|*info|*warn|*error)]

STRUCP
VERSION(yes|no)
```

Figure 5.2 Universal Copy for OS/400 - Command Line Syntax

5.4 Universal Copy for HP NonStop

This section describes Universal Copy command syntax and options for the HP NonStop operating system.

**Currently, HP NonStop runs Universal Copy 2.1.1.
This section provides information for that version.**

The Universal Copy program on the HP NonStop is strictly an OSS process. When used with UCMD Manager, the UCMD Manager's SERVER_OPTIONS option must specify a SCRIPT_TYPE option value of *OSS* (**`-server " -script_type OSS"`**) in order to indicate that this is an OSS process.

5.4.1 Configuration Options

Table 5.3, below, identifies all Universal Copy configuration options for HP NonStop.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
HELP	Prints a description of the command options and their format.
MESSAGE_LEVEL	Level of messages that will be displayed.
MODE	Mode in which input files are read and output files are written.
OUTPUT	File name that data is written to instead of standard output.
REPLACE	Specification for whether or not the existing file specified by OUTPUT is replaced.
TRANSACTIONAL	Specification for whether or not the copy operation is performed in transactional mode.
VERSION	Prints the program version and copyright information.

Table 5.3 Universal Copy Configuration Options - HP NonStop

5.4.2 Command Line Syntax

Figure 5.3, below, illustrates the syntax — using the long form of command line options — of Universal Copy for HP NonStop.

```
ucopy
[-level {trace|audit|info|warn|error}]
[-mode {binary|text}]
[-output filename [-transactional {yes|no}] [-replace {yes|no}] ]
[file ...]

ucopy
{ -version | -help }
```

Figure 5.3 Universal Copy for HP NonStop - Command Line Syntax

5.4.3 Command Operands

FILE

The file operand specifies the input files. Full or relative paths can be specified.

If no input files are specified, standard input is used.

5.5 Examples of Universal Copy

This section contains examples demonstrating the use of Universal Copy.

The following list provides a link to each example.

JCL using Universal Copy

- z/OS: Copy from z/OS Manager to Remote Windows
- z/OS: Copy from Remote Windows to z/OS Manager
- z/OS: Copy from z/OS Manager to Remote UNIX
- z/OS: Copy from Remote UNIX to z/OS Manager
- z/OS: Copy from z/OS Manager to Remote OS/400
- z/OS: Copy from Remote OS/400 to z/OS Manager
- z/OS: Copy from z/OS Manager to Remote HP NonStop
- z/OS: Copy from Remote HP NonStop to z/OS Manager
- z/OS: Third-Party Copy via z/OS Manager, from Windows to UNIX
- z/OS: Third-Party Copy via z/OS Manager, from UNIX to Windows
- z/OS: Third-Party Copy via z/OS Manager, from Windows to Windows
- z/OS: Third-Party Copy via z/OS Manager, from UNIX to UNIX
- z/OS: Copy from z/OS Manager to Remote System (in Binary)
- z/OS: Copy from Remote System to z/OS Manager (in Binary)
- z/OS: Copy from z/OS Manager to Remote z/OS (with Encryption, Compression, and Data Authentication)
- z/OS: Copy from Remote z/OS to z/OS Manager (with Encryption, Compression, and Data Authentication)
- z/OS: Copy via z/OS Manager, from Local File to Remote Windows (with Windows Date Variables)
- z/OS: Copy via z/OS Manager, from Local File to Remote UNIX (with UNIX Data Variables)

Windows

- Windows: Copy via Windows Manager, from Remote UNIX to Local Windows
- Windows: Copy via Windows Manager, From Local Windows to Remote UNIX

UNIX

- UNIX: Copy via UNIX Manager, from Remote Windows to Local UNIX
- UNIX: Copy via UNIX Manager, from Local UNIX to Remote Windows

OS/400

- OS/400: Copy via OS/400 Manager, from Remote Windows to Local OS/400
- OS/400: Copy via OS/400 Manager, from Local OS/400 to Remote Windows

Note: These examples reference the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Copy, substitute the tagged names for the untagged names. (See Section 5.3.1 Universal Products for OS/400 Commands for further information.)

HP NonStop

- HP NonStop: Copy via HP NonStop Manager, from Remote Windows to Local File
- HP NonStop: Copy via HP NonStop Manager, Local File to Remote Windows

5.5.1 z/OS: Copy from z/OS Manager to Remote Windows

Figure 5.4, below, illustrates the copying of a file from z/OS to a remote Windows system.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
@echo off
ucopy -mode text -output C:\OUTPUT.FILE
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.4 Universal Copy for z/OS - Copy from z/OS Manager to Remote Windows

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **-output** option is used with the **ucopy** command to direct the standard out to a local data set on the remote server. The **-mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.2 z/OS: Copy from Remote Windows to z/OS Manager

Figure 5.5, below, illustrates the copying of a file from a remote Windows system to z/OS.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD DISP=SHR,DSN=hlg.output.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
@echo off
ucopy -mode text C:\INPUT.FILE
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.5 Universal Copy for z/OS - Copy from Remote Windows to z/OS Manager

The JCL procedure UCMDPRC is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **-mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.3 z/OS: Copy from z/OS Manager to Remote UNIX

Figure 5.6, below, illustrates the copying of a file from z/OS to a remote UNIX system.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
/opt/universal/bin/ucopy -mode text \
 -output /usr/output.file
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
```

Figure 5.6 Universal Copy for z/OS - Copy from z/OS Manager to Remote UNIX

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **-output** option is used with the **ucopy** command to direct the standard out to a local data set on the remote server. The path to the **ucopy** binary must be specified if the directory is not defined in the user's path environmental variable. The **-mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.4 z/OS: Copy from Remote UNIX to z/OS Manager

Figure 5.7, below, illustrates the copying of a file from a remote UNIX system to z/OS.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD DISP=SHR,DSN=hlq.output.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
/opt/universal/bin/ucopy -mode text \
 /usr/input.file
//SYSIN    DD *
-s SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.7 Universal Copy for z/OS - Copy from Remote UNIX to z/OS Manager

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **-mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.5 z/OS: Copy from z/OS Manager to Remote OS/400

Figure 5.8, below, illustrates the copying of a file from z/OS to a remote OS/400 system.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
STRUCP TOFILE(LIBRARY/OUTPUTFILE)TOMBR(MEMBER)
 CPYMODE(*TEXT)
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.8 Universal Copy for z/OS - Copy from z/OS Manager to Remote OS/400

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **TOFILE** parameter is used with the **STRUCP** command to direct the standard out to a local data set on the remote server. The **CPYMODE** option is used to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.6 z/OS: Copy from Remote OS/400 to z/OS Manager

Figure 5.9, below, illustrates the copying of a file from a remote OS/400 system to z/OS.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD DISP=SHR,DSN=hlq.output.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
STRUCP FRMFILE(LIBRARY/INPUTFILE)FRMMBR(MEMBER)
 CPYMODE(*TEXT)
//SYSIN    DD *
-s SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.9 Universal Copy for z/OS - Copy from Remote OS/400 to z/OS Manager

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **CPYMODE** option is used to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.7 z/OS: Copy from z/OS Manager to Remote HP NonStop

Figure 5.10, below, illustrates the copying of a file from z/OS to a remote HP NonStop system.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
ucopy -output outputfile -mode text
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-server " -script_type OSS"
/*
```

Figure 5.10 Universal Copy for z/OS - Copy from z/OS Manager to Remote HP NonStop

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **-output** option is used with the **ucopy** command to direct the standard out to a local data set on the remote server. The **-mode text** option is used with the **ucopy** command to generate an EDIT file with a file code of 101. A value of binary (default) will generate a C file with a file code of 180. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-server " -script type OSS"	Indicates that this is an OSS process.

5.5.8 z/OS: Copy from Remote HP NonStop to z/OS Manager

Figure 5.11, below, illustrates the copying of a file from a remote HP NonStop system to z/OS.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD DISP=SHR,DSN=hlq.output.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
ucopy -mode text inputfile
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-server " -script_type OSS"
/*
```

Figure 5.11 Universal Copy for z/OS - Copy from Remote HP NonStop to z/OS Manager

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **-mode text** option is used with the **ucopy** command to read an EDIT file with a file code of 101. A value of binary (default) will read a C file with a file code of 180. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-server " -script type OSS"	Indicates that this is an OSS process.

5.5.9 z/OS: Third-Party Copy via z/OS Manager, from Windows to UNIX

Figure 5.12, below, illustrates the third-party copying of a file from a z/OS Manager, which executes a **ucopy** from Windows to UNIX.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DSN=hlq.userid(#useridunx),DISP=SHR
//LOGONDD  DD DSN=hlq.userid(#useridnt),DISP=SHR
//SCRIPT   DD *
@ECHO ON
:: TRANSFER FROM NT to UNIX
@SET UCOPYPATH=/opt/universal/bin/
@SET OUTPUTFILE=outputfile
@SET INPUTFILE=inputfile
@SET UNIXHOST=unixhost
@SET TEMPUNIXID=c:\temp\tempunixid
@SET MODE=text
ucopy -output %TEMPUNIXID%
ucmd¬
 -c " %UCOPYPATH%ucopy –output %OUTPUTFILE%"¬
 < %INPUTFILE% -host %UNIXHOST% -x %TEMPUNIXID% -l info -I -m %MODE%
SET RC=%ERRORLEVEL%
del %TEMPUNIXID%
URC %RC%
//SYSIN    DD *
-s SCRIPT
-x LOGONDD
-host NTHOST
-l info
/*
```

Figure 5.12 Universal Copy for z/OS - Third-Party Copy via z/OS, from Windows to UNIX

All informational messages will be routed to the z/OS manager. The authentication information for the UNIX server must reside on the z/OS.

The file is copied as a text file, since the default transfer mode for standard files is text.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
#USERIDUNX	Encrypted userid and password member for UNIX server
#USERIDNT	Encrypted userid and password member for NT server
UCOPYPATH	Path to UCOPY on the receiving UNIX server
OUTPUTFILE	Path and filename of receiving file on UNIX server
INPUTFILE	Path and file name of sending file on Windows server
UNIXHOST	IP address or hostname of receiving UNIX server
NTHOST	IP address or hostname of sending Windows server
TEMPUNIXID	Temporary file on the Windows server used to house the encrypted logon information for the UNIX server. This file is deleted at the bottom of the script.
MODE	Mode of file transfer (binary/text).

SYSIN Options

The SYSIN options used in this example are:

Option	Description
`-script`	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
`-host`	Directs the command to a computer with a host name of **NTHOST**.
`-x`	Specifies the DD from which to read an encrypted command options file.
`-l`	Sets the level of message information.

5.5.10 z/OS: Third-Party Copy via z/OS Manager, from UNIX to Windows

Figure 5.13, below, illustrates the third-party copying of a file from a z/OS Manager, which executes a **ucopy** from UNIX to Windows.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DSN=hlq.userid(#useridnt),DISP=SHR
//LOGONDD  DD DSN=hlq.userid(#useridunx),DISP=SHR
//SCRIPT   DD *
export UCMDPATH=/opt/universal/bin
export UCPYPATH=/opt/universal/bin
export OUTPUTFILE="c:\temp\outputfile"
export INPUTFILE=/tmp/inputfile
export NTHOST=nthostname
export TEMPNTID=/tmp/tempntid
export MODE=text
$UCPYPATH/ucopy -output $TEMPNTID
$UCMDPATH/ucmd \
 -c "ucopy –output $OUTPUTFILE"< $INPUTFILE \
 -host $NTHOST -x $TEMPNTID -l info -I -m $MODE
rc=$?
rm $TEMPNTID
exit $rc
//SYSIN    DD *
-s SCRIPT
-x LOGONDD
-host unixhostname
-l info
/*
```

Figure 5.13 Universal Copy for z/OS - Third-Party Copy via z/OS, from UNIX to Windows

All error messages will be routed to the z/OS manager. The authentication information for the NT server must reside on the z/OS.

The file is copied as a text file since the default transfer mode for standard files is text.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
#USERIDUNX	Encrypted userid and password member for UNIX server
#USERIDNT	Encrypted userid and password member for Windows server
UCOPYPATH	Path to UNIX **ucopy** executable
UCMDPATH	Path to UNIX **ucmd** executable
OUTPUTFILE	Path and filename of receiving file
INPUTFILE	Path and file name of sending file
NTHOST	IP address or hostname of receiving Windows server
TEMPNTID	Temporary file on the UNIX server used to house the encrypted logon information for the Windows server.
MODE	Mode of file transfer (binary / text). Default is set to text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
`-script`	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
`-host`	Directs the command to a computer with a host name of **`unixhostname`**.
`-x`	Specifies the DD from which to read an encrypted command options file.
`-l`	Sets the level of message information.

5.5.11 z/OS: Third-Party Copy via z/OS Manager, from Windows to Windows

Figure 5.14, below, illustrates the third-party copying of a file from a z/OS Manager, which executes a **ucopy** command from Windows to Windows.

The standard error is read into the UMET utility to verify the existence of the input file. The last step copies standard error to the job log.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DSN=hlq.userid(#nt2logon),DISP=SHR
//LOGONDD  DD DSN=hlq.userid(#ntlogon),DISP=SHR
//UNVERR   DD DSN=hlq.output(stderr),DISP=SHR
//SCRIPT   DD *
@ECHO ON
:: TRANSFER FROM NT to NT
@SET OUTPUTFILE=c:\temp\output.file
@SET INPUTFILE=c:\temp\input.file
@SET NT2HOST=hostname
@SET TEMPNT2ID=c:\temp\userid.enc
@SET MODE=text
ucopy -output %TEMPNT2ID%
ucmd¬
 -c "ucopy -output %OUTPUTFILE%" <%INPUTFILE% ¬
 -host %NT2HOST% -x %TEMPNT2ID% -l info -I -m %MODE%
SET RC=%ERRORLEVEL%
del %TEMPNT2ID%
URC %RC%
//SYSIN    DD  *
 -s SCRIPT
 -x LOGONDD
 -host NTHOST
 -l info
/*
//*
//*************************************************************
//stepname EXEC PGM=UMET,PARM='-TABLE TABLE -LEVEL VERBOSE'
//STEPLIB  DD  DISP=SHR,DSN=hlq.UNV.SUNVLOAD
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//TABLE    DD  *
  "The system cannot find the file specified." 8
```

```
/*
//SYSIN    DD  DISP=SHR,DSN=hlq.output(stderr)
//*************************************************************
//stepname EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=hlq.output(stderr)
//SYSUT2   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD DUMMY
```

Figure 5.14 Universal Copy for z/OS - Third-Party Copy via z/OS, from Windows to Windows

All error messages will be routed to the z/OS manager. The authentication information for the WIndows server must reside on the z/OS.

The file is copied as a text file, since the default transfer mode for standard files is text.

The UMETSTEP step executes the UMET utility. UMET is used to set the condition code field to a value based on message text. The SYSIN DD is the standard error of the first step and the TABLE DD is the table defining which condition code to be used when text is found.

Note: The UMET program is used because native Windows returns a 0 return (exit) code, even when the stdin does not exist. Therefore, the process would end with a 0, even if the input file did not exist. UMET will set the condition code to 8.

The IEBGENER step will copy the standard error file to SYSLOG if the process gets a non-zero condition code.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
#USERIDNT	Encrypted userid and password member for sending Windows server
#USERIDNT2	Encrypted userid and password member for receiving Windows server
OUTPUTFILE	Path and filename of receiving file
INPUTFILE	Path and file name of sending file
NTHOST	IP address or hostname of sending Windows server
NT2HOST	IP address or hostname of receiving Windows server
TEMPNT2ID	Temporary file on the Windows sending server used to house the encrypted logon information for the Windows receiving server.
MODE	Mode of file transfer (binary / text). Default is set to text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
`-script`	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
`-host`	Directs the command to a computer with a host name of **NTHOST**.
`-x`	Specifies the DD from which to read an encrypted command options file.
`-l`	Sets the level of message information.

5.5.12 z/OS: Third-Party Copy via z/OS Manager, from UNIX to UNIX

Figure 5.15, below, illustrates the third-party copying of a file from a z/OS Manager, which executes a **ucopy** command from UNIX to UNIX.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DSN=hlq.userid(useridunxr),DISP=SHR
//LOGONDD  DD DSN=hlq.userid(useridunxs),DISP=SHR
//SCRIPT   DD *
export UCOPYPATH=/opt/universal/bin
export UCMDPATH=/opt/universal/bin
export OUTPUTFILE=/outputfile
export INPUTFILE=/inputfile
export UNIXRHOST=receivinghostname
export TEMPUNIXRID=/tmp/unixid.tmp
export MODE=text
$UCOPYPATH/ucopy –output $TEMPUNIXRID
$UCMDPATH/ucmd \
 -c "$UCOPYPATH/ucopy –output $OUTPUTFILE" < $INPUTFILE \
-host $UNIXRHOST -x $TEMPUNIXRID -l info -I -m $MODE
rc=$?
rm $TEMPUNIXRID
exit $rc
//SYSIN    DD *
 -s SCRIPT
 -x LOGONDD
 -host unixshost
 -l info
/*
```

Figure 5.15 Universal Copy for z/OS - Third-Party Copy via z/OS, from UNIX to UNIX

All error messages will be routed to the z/OS manager. The authentication information for both UNIX servers must reside on the z/OS.

The file is copied as a text file since the default transfer mode for standard files is text.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
UCOPYPATH	Path pointing to the **ucopy** executable on the second UNIX server
UCMDPATH	Path pointing to the **ucmd** executable on the second UNIX server
OUTPUTFILE	Path and filename of receiving file
INPUTFILE	Path and file name of sending file
UNIXSHOST	IP address or hostname of sending UNIX server
UNIXRHOST	IP address or hostname of receiving UNIX server
TEMPUNIXRID	Temporary file on the sending UNIX server used to house the encrypted logon information for the receiving UNIX server.
MODE	Mode of file transfer (binary / text). Default is set to text.
USERUNXR	Should point to the userid/password information for the receiving UNIX server
USERUNXS	Should point to the userid/password information for the sending UNIX server

SYSIN Options

The SYSIN options used in this example are:

Option	Description
`-script`	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
`-host`	Directs the command to a computer with a host name of **`unixshost`**.
`-x`	Specifies the DD from which to read an encrypted command options file.
`-l`	Sets the level of message information.

5.5.13 z/OS: Copy from z/OS Manager to Remote System (in Binary)

Figure 5.16, below, illustrates the copying of a file from z/OS manager to a remote system, in binary, with no end-of-line character interpretation.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
@echo off
ucopy -output C:\OUTPUT.FILE
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-stdin –mode binary
/*
```

Figure 5.16 Universal Copy for z/OS - Copy from z/OS Manager to Remote System (in Binary)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **–output** option used with the **ucopy** command directs the stdout to a local data set on the remote server. The **-mode** option used with the **ucopy** command defaults to binary, so no end-of-line character interpretation is done. Binary is specified for standard input transfer mode. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-stdin	Specifies that the options following this one apply to the stdin file.
-mode	Specifies whether transferred data is treated as text or binary.

5.5.14 z/OS: Copy from Remote System to z/OS Manager (in Binary)

Figure 5.17, below, illustrates the copying of a file from a remote system to z/OS Manager, in binary, with no end-of-line character interpretation.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD DISP=SHR,DSN=hlg.output.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
@echo off
ucopy C:\INPUT.FILE
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-stdout –mode binary
/*
```

Figure 5.17 Universal Copy for z/OS - Copy from Remote System to z/OS Manager (in Binary)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **-mode** option used with the **ucopy** command defaults to binary, so no end-of-line character interpretation is done. Binary is specified for standard output transfer mode. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-stdout	Specifies that the options following this one apply to the stdout file.
-mode	Specifies whether transferred data is treated as text or binary.

5.5.15 z/OS: Copy from z/OS Manager to Remote z/OS (with Encryption, Compression, and Data Authentication)

Figure 5.18, below, illustrates the copying of a file from z/OS Manager to a remote z/OS system (with encryption, compression, and data authentication).

```
//STEP1 EXEC UCMDPRC
//UNVIN='DISP=SHR,DSN=MY.PDS(MEMBER)
//LOGONDD  DD DISP=SHR,DSN=MY.LOGON(USERID)
//SCRIPTDD DD *
 /opt/universal/bin/ucopy > //'REMOTE.PDS(MEMBER)'
//SYSIN DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-stdin -e yes -k yes -a yes
/*
```

Figure 5.18 Universal Copy for z/OS - Copy from z/OS Manager to Remote z/OS (with Encryption, Compression, and Data Authentication)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **–output** option is used with the **ucopy** command to direct the standard out to a local data set on the remote server. The **–mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-stdin	Specifies that the options following this one apply to the stdin file.
-e	Specifies that standard file data sent over the network is encrypted.
-k	Specifies whether the standard file data transmitted across the network should be compressed.
-a	Specifies that the standard file data sent over the network is authenticated.

5.5.16 z/OS: Copy from Remote z/OS to z/OS Manager (with Encryption, Compression, and Data Authentication)

Figure 5.19, below, illustrates the copying of a file from a remote z/OS system to z/OS Manager (with encryption, compression, and data authentication).

```
//STEP1 EXEC UCMDPRC
//UNVOUT='DISP=SHR,DSN=MY.PDS(MEMBER)
//LOGONDD  DD DISP=SHR,DSN=MY.LOGON(USERID)
//SCRIPTDD DD *
 /opt/universal/bin/ucopy < //'REMOTE.PDS(MEMBER)'
//SYSIN DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
-stdout -e yes -k yes -a yes
/*
```

Figure 5.19 Universal Copy for z/OS - Copy from Remote z/OS to z/OS Manager (with Encryption, Compression, and Data Authentication)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD specifies a local data set to use for the standard output of the remote command. The **–mode text** option is used with the **ucopy** command to force end-of-line character interpretation. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.
-stdout	Specifies that the options following this one apply to the stdout file.
-e	Specifies that standard file data sent over the network is encrypted.
-k	Specifies whether the standard file data transmitted across the network should be compressed.
-a	Specifies that the standard file data sent over the network is authenticated.

5.5.17 z/OS: Copy via z/OS Manager, from Local File to Remote Windows (with Windows Date Variables)

Figure 5.20, below, illustrates the copying of a file from z/OS to a remote Windows system.

The file name on the Windows server is dynamically created based on the current date.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
@echo off
for /f "tokens=1 delims=/" %%a in ('date /t') do set daymm=%%a
for /f "tokens=2" %%a in ('echo %daymm%') do set mm=%%a
for /f "tokens=2 delims=/" %%a in ('date /t') do set dd=%%a
for /f "tokens=3 delims=/" %%a in ('date /t') do set yy=%%a
echo daymm:  %daymm%
echo mmddyy: %mm%%dd%%yy%
ucopy -output c:\temp\outputfile%mm%%dd%%yy%
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
/*
```

Figure 5.20 Universal Copy for z/OS - Copy from Local File to Remote Windows (with Windows Date Variables)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The **-output** option is used with the **ucopy** command to direct stdout to a local data set on the remote Windows server. The file name is created with a date variable. The date variable is set to the current date in the commands preceding the **ucopy** command. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

The file is copied as a text file since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.18 z/OS: Copy via z/OS Manager, from Local File to Remote UNIX (with UNIX Data Variables)

Figure 5.21, below, illustrates the copying of a file from z/OS to a remote UNIX system.

The file name on the UNIX server is dynamically created based on the current date.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.input.file
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
DATEN=`date +%d%m`
export DATEN
echo $DATEN
/opt/universal/bin/ucopy \
 -output /tmp/output$DATEN.file
//SYSIN    DD *
-script SCRIPTDD
-x LOGONDD
-host dallas
```

Figure 5.21 Universal Copy for z/OS - Copy from Local File to Remote UNIX (with UNIX Date Variables)

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to a remote system named **dallas** for execution. The stdout redirection character **>** is used with the **ucopy** command to direct stdout to a local data set on the remote server. The file name is created with a date variable. The date variable is set to the current date in the commands preceding the **ucopy** command. The path to the **ucopy** binary must be specified if the directory is not defined in the user's path environmental variable. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**. The UNIX continuation character **** is used to split the **ucopy** command to two lines.

The file is copied as a text file, since the default transfer mode for standard files is text.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **dallas**.
-x	Specifies the DD from which to read an encrypted command options file.

5.5.19 Windows: Copy via Windows Manager, from Remote UNIX to Local Windows

Figure 5.22, below, illustrates the copying of a file from a remote UNIX system to a local Windows system.

Although the command is shown on two lines, it should be entered on one line at the command prompt. If it is coded in a script, the Windows continuation character of ¬ must be used.

```
ucmd -cmd " /opt/universal/bin/ucopy unixinputfile"
-host unixhost -x unixid.file > c:\temp\ntoutputfile
```

Figure 5.22 Universal Copy for Windows - Copy from Remote UNIX to Local Windows

The standard out of the **ucopy** command on the remote host is redirected back to the local host and written to **c:\temp\ntoutput.file**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file, since the default transfer mode for standard files is text.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
ntoutputfile	Path and filename of output file
unixinputfile	Path and file name of input file
unixhost	IP address of remote UNIX server
unixid.file	File on the Windows server used to house the authentication parameters for the UNIX server

Command Line Options

The command line options used in this example are:

Option	Description
-cmd	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its STDOUT.
-host	Directs the command to a computer with a host name of **unixhost**.
-x	Specifies the file from which to read encrypted command options.

5.5.20 Windows: Copy via Windows Manager, From Local Windows to Remote UNIX

Figure 5.23, below, illustrates the copying of a file from a local Windows system to a remote UNIX system.

Although the command is shown on two lines, it should be entered on one line at the command prompt. If it is coded in a script, the Windows continuation character of ¬ must be used.

```
ucmd -cmd " /opt/universal/bin/ucopy -output /tmp/unixoutputfile"
-host unixhost -x unixid.file
< c:\temp\ntinputfile
```

Figure 5.23 Universal Copy for Windows - Copy from Local Windows to Remote UNIX

The stdin of the **ucmd** manager on the local host is redirected to the stdout of the remote host and written to **/tmp/unixoutputfile**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file since the default transfer mode for standard files is text.

Parameters

The following parameters should be changed to match your information:

Parameter	Description
unixoutputfile	Path and filename of output file
ntinputfile	Path and file name of input file
unixhost	IP address of sending UNIX server
unixid.file	File on the Windows server used to house the authentication parameters for the UNIX server

Command Line Options

The command line options used in this example are:

Option	Description
-cmd	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its stdout.
-host	Directs the command to a computer with a host name of **unixhost**.
-x	Specifies the file from which to read encrypted command options.

5.5.21 UNIX: Copy via UNIX Manager, from Remote Windows to Local UNIX

Figure 5.24, below, illustrates the copying of a file from a remote Windows system to a local UNIX server.

If it is coded in a script, then the UNIX continuation character of \ must be used.

```
ucmd -cmd 'ucopy ntinputfile' -host nthost -x ntid.file > /tmp/unixoutputfile
```

Figure 5.24 Universal Copy for UNIX - Copy from Remote Windows to Local UNIX

The stdout of the **ucopy** command on the remote host is redirected back to the local host and written to **/tmp/unixoutputfile**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file, since the default transfer mode for standard files is text.

Command Line Options

The command line options used are:

Option	Description
-cmd	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its stdout.
-host	Directs the command to a computer with a host name of **nthost**.
-x	Specifies the file from which to read encrypted command options.

5.5.22 UNIX: Copy via UNIX Manager, from Local UNIX to Remote Windows

Figure 5.25, below, illustrates the copying of a file from a local UNIX system to a remote Windows server.

Although the command is shown on two lines, it should be entered on one line at the command prompt. If it is coded in a script, the UNIX continuation character of \ must be used.

```
ucmd -cmd 'ucopy -output c:\temp\ntoutput.file' -host nthost
-x login.file < /tmp/unixinput.file
```

Figure 5.25 Universal Copy for UNIX - Copy from Local UNIX to Remote Windows

The stdin of the ucmd manager on the local host is redirected to the remote host and written to stdout file **c:\temp\ntoutput.file**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file, since the default transfer mode for standard files is text.

Command Line Options

The command line options used are:

Option	Description
-cmd	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its stdout.
-host	Directs the command to a computer with a host name of **nthost**.
-x	Specifies the file from which to read encrypted command options.

5.5.23 OS/400: Copy via OS/400 Manager, from Remote Windows to Local OS/400

Figure 5.26, below, illustrates the copying of a file from a remote Windows system to a local OS/400 file.

```
STRUCM CMD('ucopy c:\ntinput.file') HOST(nthost) USERID(joe) PWD(akkSdiq)
SOTFILE(library/outputfile) SOTMBR(member)
```

Figure 5.26 Universal Copy for OS/400 - Copy from Remote Windows to Local OS/400

The stdout of the **ucopy** command on the remote host is redirected back to the local host and written to the stdout of ucmd, which is directed to the local file **SOTFILE** and, optionally, **SOTMBR**.

The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file, since the default transfer mode for standard files is text.

Command Line Options

The command line options used are:

Option	Description
CMD	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its stdout.
HOST	Directs the command to a computer with a host name of **nthost**.
USERID	Specifies the remote user ID with which to execute the command.
PWD	Specifies the password for the user ID.

5.5.24 OS/400: Copy via OS/400 Manager, from Local OS/400 to Remote Windows

Figure 5.27, below, illustrates the copying of a local OS/400 file to a remote Windows system.

```
STRUCM CMD('ucopy -output c:\ntoutput.file') HOST(nthost) USERID(joe)
PWD(akksdiq) SINFILE(library/inputfile) SINMBR(member)
```

Figure 5.27 Universal Copy for OS/400 - Copy from Local OS/400 to Remote Windows

The **ucopy** command receives its stdin file from ucmd. The stdin of ucmd is redirected from **SINFILE** and, optionally, **SINMBR** to stdout of **ucopy**, which is **c:\ntoutput.file**.

The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file, since the default transfer mode for standard files is text.

Command Line Options

The command line options used are:

Option	Description
CMD	Specifies the remote command **ucopy > file** to execute. The **ucopy** program copies its stdin to its stdout.
HOST	Directs the command to a computer with a host name of **nthost**.
USERID	Specifies the remote user ID with which to execute the command.
PWD	Specifies the password for the user ID.

5.5.25 HP NonStop: Copy via HP NonStop Manager, from Remote Windows to Local File

Figure 5.28, below, illustrates the copying of a file from a remote Windows system to a local file.

Although the command shown is on two lines, it should be entered on one line at the command prompt.

The HP NonStop manager is executed within the TACL environment.

```
run $SYSTEM.UNVBIN.ucmd /OUT outputfile/ -cmd 'ucopy inputfile' -host dallas
-userid joe -pwd akkSdiq
```

Figure 5.28 Universal Copy for HP NonStop - Copy from Remote Windows to Local File

The stdout of the **ucopy** command on the remote host is redirected back to the local host and written to the stdout of ucmd, which is then redirected to the local file **outputfile**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

The file is copied as a text file since the default transfer mode for standard files is text.

Command Line Options

The command line options used are:

Option	Description
-cmd	Specifies the remote command **ucopy file** to execute. The **ucopy** program copies the files specified on the command line to its stdout.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Specifies the remote user ID with which to execute the command.
-pwd	Specifies the password for the user ID.

5.5.26 HP NonStop: Copy via HP NonStop Manager, Local File to Remote Windows

Figure 5.29, below, illustrates the copying of a local file to a remote Windows system.

Although the command is shown on two lines, it should be entered on one line at the command prompt.

The HP NonStop manager is executed within the TACL environment.

The file is copied as a text file, since the default transfer mode for standard files is text.

```
run $SYSTEM.UNVBIN.ucmd /IN inputfile/ –cmd 'ucopy –output outputfile'
-host dallas -userid joe –pwd akkSdiq
```

Figure 5.29 Universal Copy for HP NonStop - Copy Local File to Remote Windows

The **ucopy** command receives its stdin file from ucmd. The stdin of ucmd is redirected from **inputfile**. The command **ucopy** is installed as part of Universal Command Server on the remote system.

Command Line Options

The command line options used are:

Option	Description
-cmd	Specifies the remote command **ucopy** to execute. The **ucopy** program copies stdin to stdout.
-host	Directs the command to a computer with a host name of **dallas**.
-userid	Specifies the remote user ID with which to execute the command.
-pwd	Specifies the password for the user ID.

Chapter 6
Universal Database Dump

6.1 Overview

Universal Products databases are implemented using Oracle's Berkeley Database product. The Berkeley Database provides utilities to perform administrative database tasks.

The Universal Database Dump (UDBDUMP) utility is the Berkeley **db_dump** utility tailored specifically for Universal Products databases.

UDBDUMP and the Universal Database Load (UDBLOAD) utility (see Chapter 7 Universal Database Load) are provided to enable recovery from a corrupted Berkeley database. Databases can potentially become corrupt due to system and address spaces ending abnormally.

Oracle documentation on **db_dump** and all other utility commands is provided at the following URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

6.1.1 Usage

UDBDUMP invokes the Berkeley **db_dump** utility. The UDBDUMP command line options are passed to **db_dump**. UDBDUMP reads a specified database file and dumps the contents to a database dump file.

This dump file can be loaded into a database using UDBLOAD.

6.2 Universal Database Dump for z/OS

This section describes Universal Database Dump (UDBDUMP) utility, specific to the IBM z/OS operating system.

6.2.1 JCL Procedure

Figure 6.1, below, illustrates the Universal Database Dump for z/OS JCL procedure (**UDBDPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//UDBLPRC  PROC DBFILE=,
//             SHLQ=#SHLQ,
//             DMPDSN=,
//             DBHFS=
//*
//S1       EXEC PGM=UDBDUMP,
// PARM='ENVAR(TZ=EST5EDT)/&DBHFS -r &DBFILE'
//STEPLIB  DD  DSN=&SHLQ..UNV.SUNVLOAD,
//             DISP=SHR
//*
//UNVOUT   DD  DSN=&DMPDSN,
//             DISP=SHR
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DUMMY
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//         PEND
```

Figure 6.1 Universal Database Dump for z/OS – JCL Procedure

6.2.2 DD Statements used in JCL Procedure

Table 6.1, below, describes the DD statements used in the Universal Database Dump for z/OS JCL procedure illustrated in Figure 6.1.

ddname	Description
STEPLIB	Load library in which program UDBDUMP program is located.
UNVOUT	Database dump file. The dump data set must be a physically sequential data set with a variable-block record format, a record length of 32756, and a block size of 32760.
SYSPRINT	UDBDUMP standard output ddname.
SYSOUT	UDBDUMP standard error ddname.
SYSIN	UDBDUMP standard input.

Table 6.1 Universal Database Dump for z/OS – DD Statements in JCL Procedure

6.2.3 JCL

Figure 6.2, below, illustrates the Universal Database Dump for z/OS JCL.

```
//STEP1    EXEC PGM=UDBDUMP,PARM='options database'
//STEPLIB  DD  DSN=UNV.SUNVLOAD,DISP=SHR
//*
//UNVOUT   DD  DSN=DB.DUMP,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DUMMY
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

Figure 6.2 Universal Database Dump for z/OS – JCL

6.2.4 Configuration Options

Table 6.2 identifies the UDBDUMP for z/OS configuration options. It describes only those options relevant for database recovery.

For details on all options, see the Oracle documentation on the **db_dump** utility at URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
DATABASE_FILE	Database file to be dumped.
RECOVER	Specification to recover as many records as possible from a possibly corrupt database file.

Table 6.2 Universal Database Dump for z/OS - Configuration Options

Note: UDBDUMP accepts configuration options only on the PARM keyword of the **EXEC** statement.

6.2.5 Command Line Syntax

Figure 6.4, below, illustrates the command line syntax of UDBDUMP for z/OS. It identifies only those options that are relevant for database recovery.

```
[-r]
database
```

Figure 6.3 Universal Database Dump for z/OS - Command Line Syntax

6.3 Universal Database Dump for Windows and UNIX

This section describes Universal Database Dump (UDBDUMP) utility, specific to the Windows and UNIX operating systems.

6.3.1 Configuration Options

Table 6.3 identifies the UDBDUMP for Windows and UNIX configuration options. It describes only those options relevant for database recovery.

For details on all options, see the Oracle documentation on the **db_dump** utility at URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
DATABASE_FILE	Database file to be dumped.
RECOVER	Specification to recover as many records as possible from a possibly corrupt database file.

Table 6.3 Universal Database Dump for Windows UNIX - Configuration Options

6.3.2 Command Line Syntax

Figure 6.4, below, illustrates the syntax of UDBDUMP for Windows and UNIX. It identifies only those options that are relevant for database recovery.

```
udb_dump
[-r]
database > dump
```

Figure 6.4 Universal Database Dump for Windows and UNIX - Command Line Syntax

Chapter 7
Universal Database Load

7.1 Overview

Universal Products databases are implemented using Oracle's Berkeley Database product. The Berkeley Database provides utilities to perform administrative database tasks.

The Universal Database Load (UDBLOAD) utility is the Berkeley **db_load** utility tailored specifically for Universal Product databases.

UDBLOAD and the Universal Database Dump (UDBDUMP) utility (see Chapter 6 Universal Database Dump) are provided to enable recovery from a corrupted Berkeley databases. Databases can potentially become corrupt due to system and address spaces ending abnormally.

Oracle documentation on **db_load** and all other utility commands is provided at the following URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

7.1.1 Usage

UDBLOAD invokes the Berkeley **db_load** utility. The UDBLOAD command line options are passed to **db_load**. UDBLOAD reads the database dump file and loads the contents into the specified database file.

The database dump file is created with UDBDUMP.

Note: By default, the load operation updates the specified database file; it does not overwrite it. To overwrite the file, the OVERWRITE option must be specified.

You should back up the database file prior to performing any load operation.

7.2 Universal Database Load for z/OS

This section describes Universal Database Load (UDBLOAD) utility, specific to the IBM z/OS operating system.

7.2.1 JCL Procedure

Figure 7.1, below, illustrates the Universal Database Load for z/OS JCL procedure (**UDBLPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//UDBLPRC  PROC DBFILE=,
//             SHLQ=#SHLQ,
//             DMPDSN=,
//             DBHFS=
//*
//S1       EXEC PGM=UDBLOAD,
//  PARM='ENVAR(TZ=EST5EDT)/&DBHFS -o &DBFILE'
//STEPLIB  DD  DSN=&SHLQ..UNV.SUNVLOAD,
//             DISP=SHR
//*
//UNVIN    DD  DSN=&DMPDSN,
//             DISP=SHR
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DUMMY
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//         PEND
```

Figure 7.1 Universal Database Load for z/OS – JCL Procedure

7.2.2 DD Statements used in JCL Procedure

Table 7.1, below, describes the DD statements used in the Universal Database Load for z/OS JCL illustrated in Figure 7.1.

ddname	Description
STEPLIB	Load library in which program UDBLOAD program is located.
UNVIN	Database dump file produced by Universal Database Dump.
SYSPRINT	UDBLOAD standard output ddname.
SYSOUT	UDBLOAD standard error ddname.
SYSIN	UDBLOAD standard input.

Table 7.1 Universal Database Load for z/OS – DD Statements in JCL Procedure

7.2.3 JCL

Figure 7.2, below, illustrates the Universal Database Load for z/OS JCL.

```
//STEP1    EXEC PGM=UDBLOAD,PARM='options database'
//STEPLIB  DD  DSN=UNV.SUNVLOAD,DISP=SHR
//*
//UNVIN    DD  DSN=db.dump,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DUMMY
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

Figure 7.2 Universal Database Load for z/OS – JCL

7.2.4 Configuration Options

Table 7.2 identifies the UDBLOAD for z/OS configuration options. It describes only those options relevant for database recovery.

For details on all options, see the Oracle documentation on the **db_dump** utility at URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
DATABASE_FILE	Database file to be loaded.
OVERWRITE	Specification to overwrite the database file, not update it. (Update is the default load operation.)

Table 7.2 Universal Database Load for z/OS - Configuration Options

Note: UDBLOAD accepts configuration options only on the PARM keyword of the **EXEC** statement.

7.2.5 Command Line Syntax

Figure 7.4, below, illustrates the command line syntax of UDBLOAD for z/OS. It identifies only those options that are relevant for database recovery.

```
[-o]
database
```

Figure 7.3 Universal Database Load for z/OS - Command Line Syntax

7.3 Universal Database Load for Windows and UNIX

This section describes Universal Database Dump (UDBLOAD) utility, specific to the Windows and UNIX operating systems.

7.3.1 Configuration Options

Table 7.3 identifies the UDBLOA for Windows and UNIX configuration options. It describes only those options relevant for database recovery.

For details on all options, see the Oracle documentation on the **db_dump** utility at URL:

http://www.oracle.com/technology/documentation/berkeley-db/db/index.html

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
DATABASE_FILE	Database file to be loaded.
OVERWRITE	Specification to overwrite the database file, not update it. (Update is the default load operation.)

Table 7.3 Universal Database Load for Windows and UNIX - Configuration Options

7.3.2 Command Line Syntax

Figure 7.4, below, illustrates the syntax of UDBLOAD for Windows and UNIX. It identifies only those options that are relevant for database recovery.

```
udb_load
[-o]
database < dump
```

Figure 7.4 Universal Database Load for Windows and UNIX - Command Line Syntax

Chapter 8
Universal Display Log File

8.1 Overview

Universal Display Log File (**UDSPLOGF**) is a command for the OS/400 environment.

Universal Display Log File reads job log output files that were created as a result of API **QMHCTLJL** or command **DSPJOBLOG**. The job log is formatted and written to standard output.

Optionally, **UDSPLOGF** can delete the job log file members after writing. File member deletion is controlled by the REMOVE_MEMBERS option. The default behavior is to leave the members unaltered.

Note: Universal Display Log File became available for the OS/400 environment with PTF 0UC0114 (maintenance level 1.2.1.10).

8.2 Usage

Universal Display Log File consists of a command line program followed by a list of configuration options.

This section describes the configuration options and their command line syntax.

8.2.1 Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This chapter references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Display Log File, substitute the tagged names for the untagged names in these references.

8.2.2 Configuration Options

Table 8.1, below, identifies all Universal Display Log configuration options.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
PRIMARY_FILE	Name of the primary output file.
PRIMARY_MEMBER	Name of the primary output file member.
REMOVE_MEMBERS	Controls the deletion of job log output file members.
SECONDARY_FILE	Name of the secondary output file.
SECONDARY_MEMBER	Name of the secondary output file member.

Table 8.1 Universal Display Log File - Configuration Options

8.2.3 Command Line Syntax

Figure 8.1, below, illustrates the command line syntax of Universal Display Log File.

```
UDSPLOGF
[PRMRYFILE(filename[library]) [PRMRYMBR(member)] ]
[SCNDRYFILE(filename[library]) [SCNDRYMBR(member)] ]
[REMOVE({yes|no})]
```

Figure 8.1 Universal Display Log File - Command Line Syntax

Chapter 9
Universal Encrypt

9.1 Overview

The Universal Encrypt (**UENCRYPT**) utility encrypts the contents of command files into an unintelligible format for privacy reasons.

Universal Product programs have the ability to read command line arguments from command files. Command files can be used to save private information, such as user identifiers and passwords. These files can be stored as clear text or encrypted text.

Command files that contain private information must be protected by using local file system security. This ensures that only authorized accounts have read access, regardless of whether or not the command files are encrypted.

Universal Encrypt adds an additional layer of security by encrypting the contents of command files. However, it should not be mistaken as a replacement for file system security.

The encrypted command file can be decrypted only by Universal Product programs. No decrypt command is provided to decrypt the command file.

9.1.1 Usage

Universal Encrypt reads a command file from its standard input and writes an encrypted command file to its standard output.

The encrypted command file is a text file that can be used on any operating system by any Universal Product. Lines starting with a hash (**#**) character in column one are comments; blank lines are ignored.

Universal Encrypt performs operations specified by the command options.

9.2 Universal Encrypt for z/OS

This section describes Universal Encrypt program JCL, configuration options, and command line syntax for the z/OS operating system.

9.2.1 JCL

Figure 9.1, below, illustrates the Universal Encrypt for z/OS JCL.

```
//UENCRYPT EXEC PGM=UENCRYPT
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//UNVIN    DD  DISP=SHR,DSN=MY.CLEAN.CMDFILE
//UNVOUT   DD  DISP=SHR,DSN=MY.ENCRYPT.CMDFILE
//SYSIN    DD  *
  -KEY DF#98AD@ -AES YES,/*
```

Figure 9.1 Universal Encrypt for z/OS – JCL

9.2.2 DD Statements used in JCL

Table 9.1, below, describes the DD statements used in the Universal Encrypt for z/OS JCL illustrated in Figure 9.1.

ddname	Description
STEPLIB	Load library in which program UENCRYPT is located.
SYSPRINT	UENCRYPT standard output ddname.
SYSOUT	UENCRYPT standard error ddname.
UNVIN	Clear text command file to encrypt.
UNVOUT	Encrypted command file.
SYSIN	UENCRYPT standard input from which parameters are read.

Table 9.1 Universal Encrypt for z/OS – DD Statements in JCL

9.2.3 Configuration

Universal Encrypt operations are controlled by the configuration options specified either on the PARM keyword of the **EXEC** statement or in the SYSIN ddname.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

9.2.4 Configuration Options

Table 9.2, below, identifies the Universal Encrypt for z/OS configuration options.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
AES	Specification for whether or not AES encryption is used.
CODE_PAGE	Character code page used to translate text data received and encrypted.
ENCRYPTION_KEY	Encryption key used by the encryption algorithm.
HELP	Prints a description of the command options and their format.
VERSION	Prints the program version and copyright information.

Table 9.2 Universal Encrypt for z/OS - Configuration Options

9.2.5 Command Line Syntax

Figure 9.2 illustrates the command line syntax – using the long form of command line options – of Universal Encrypt for z/OS.

```
[-key key]
[-codepage codepage]
[-aes {yes|no}]

uencrypt
{ -help | -version }
```

Figure 9.2 Universal Encrypt for z/OS - Command Line Syntax

9.2.6 Example

Assume that a command file named **MY.CLEAR.CMDFILE** contains the following data:

```
-u Thomas –w thames
```

Figure 9.3 Universal Encrypt for z/OS - Clear Command File

The following JCL encrypts the command file allocated to ddname **UNVIN** using AES encryption and an encryption key **MYKEY123**:

```
//UENCRYPT EXEC PGM=UENCRYPT
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//UNVIN    DD  DISP=SHR,MY.CLEAR.CMDFILE
//UNVOUT   DD  DISP=SHR,MY.ENCRYPT.CMDFILE
//SYSIN    DD  *
  -KEY MYKEY123 –AES YES
/*
```

Figure 9.4 Universal Encrypt for z/OS - JCL

The resulting encrypted command file is written to ddname **UNVOUT**.

Figure 9.5, below, illustrates the contents of **MY.ENCRYPT.CMDFILE**.

```
# Universal Encrypt
#   Date . . . . . : Thu Nov  3 07:29:03 2005
#   User . . . . . : T02JAH1
#   Host . . . . . : hosta.acme.com
#   Program  . . . : uencrypt 3.2.0 Level 5 Release Build 130
#   Encryption . . : AES 256-bit

1F7DAF62583C813EA874CA168FF626C348F7BF171477D380D9A2FFFED33C539B71B4206EA5021F
D92CDFDD931C3B88B9CD711A4693EFE6B49FAE9431E9C946F7F35C9B4C31335BFB3F97F0686EFF
37068245A6B58CBFE2ADE32997A132C4114AC52CD615B2E7E8672ED0BF9867CA13B1
```

Figure 9.5 Encrypted Command File

This encrypted command file can now be used by any Universal Product command on any platform by specifying the encryption key MYKEY123.

9.3 Universal Encrypt for Windows and UNIX

This section describes Universal Encrypt configuration options and command line syntax for the Windows and UNIX operating systems.

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

9.3.1 Configuration Options

Table 9.3 identifies the Universal Encrypt for Windows and UNIX configuration options.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
AES	Specification for whether or not AES encryption is used.
CODE_PAGE	Character code page used to translate text data received and encrypted.
ENCRYPTION_KEY	Encryption key used by the encryption algorithm.
HELP	Writes a description of the command options and their format.
NLS_DIRECTORY	Directory name where Universal Encrypt can find its code page tables.
VERSION	Writes the program version and copyright information.

Table 9.3 Universal Encrypt for Windows and UNIX - Configuration Options

9.3.2 Command Line Syntax

Figure 9.6, below, illustrates the command line syntax – using the long form of configuration options – of Universal Encrypt for Windows and UNIX.

```
uencrypt
[-key key]
[-codepage codepage]
[-nlsdir directory]
[-aes {yes|no}]

uencrypt
{ -help | -version }
```

Figure 9.6 Universal Encrypt for Windows and UNIX - Command Line Syntax

9.3.3 Example

Assume that a command file named **cmdfile.clear** contains the following data:

```
-u Thomas –w thames
```

Figure 9.7 Universal Encrypt for Windows and UNIX - Contents of Command File (Sample)

The following command encrypts the command file using AES encryption with an encryption key **MYKEY123**.

```
uencrypt –key MYKEY123 -aes yes <cmdfile.clear >cmdfile.encrypt
```

Figure 9.8 Universal Encrypt for UNIX and Windows - Command Syntax (Sample)

The resulting encrypted command file is written to file **cmdfile.encrypt**.

Figure 9.9, below, illustrates the contents of **cmdfile.encrypt**.

```
# Universal Encrypt
#   Date . . . . . : Thu Nov  3 07:29:03 2005
#   User . . . . . : T02JAH1
#   Host . . . . . : hosta.acme.com
#   Program  . . . : uencrypt 3.2.0 Level 5 Release Build 130
#   Encryption . . : AES 256-bit

1F7DAF62583C813EA874CA168FF626C348F7BF171477D380D9A2FFFED33C539B71B4206EA502
1FD92CDFDD931C3B88B9CD711A4693EFE6B49FAE9431E9C946F7F35C9B4C31335BFB3F97F068
6EFF37068245A6B58CBFE2ADE32997A132C4114AC52CD615B2E7E8672ED0BF9867CA13B1
```

Figure 9.9 Universal Encrypt for Windows and UNIX - Encrypted Command File

This encrypted command file now can be used by any Universal Product command, on any operating system, by specifying the encryption key **MYKEY123**.

9.4 Universal Encrypt for OS/400

This section describes Universal Encrypt program configuration options and command line syntax for the OS/400 operating system.

9.4.1 Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This section references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Encrypt, substitute the tagged names for the untagged names in these references.

9.4.2 Configuration Options

Table 9.4 identifies the Universal Encrypt for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
AES	Specification for whether or not AES encryption is used.
CODE_PAGE	Character code page used to translate text data.
ENCRYPTION_KEY	Encryption key used to encrypt the input file.
INPUT_FILE	Input file that is to be encrypted.
OUTPUT_FILE	File to which the encrypted input file is written.
VERSION	Writes the program version information and copyright.

Table 9.4 Universal Encrypt for OS/400 - Configuration Options

9.4.3 Command Line Syntax

Figure 9.10 illustrates the command line syntax – using the STRUEN parameter form of configuration options – of Universal Encrypt for OS/400.

```
STRUEN
[INFILE(input_file) [INMBR(member)] ]
[OUTFILE(output_file) [OUTMBR(member)] ]
[KEY(key)]
[AES(*{yes|no})]
[CODEPAGE(codepage)]

STRUEN
VERSION({yes|no})
```

Figure 9.10 Universal Encrypt for OS/400 - Command Line Syntax

Note: Options entered into plain text files or encrypted files must be in short form or long form syntax (see COMMAND_FILE_PLAIN and COMMAND_FILE_ENCRYPTED in the Universal Command 3.2.0 Reference Guide.)

9.4.4 Example

Assume that a command file named **MYLIB/QTXTSRC(TESTLOGIN)** contains the following data:

```
-u Thomas -w tz74gan
```

Figure 9.11 Universal Encrypt for OS/400 - Contents of Sample Command File

The following command encrypts the command file using non-AES encryption with an encryption key **MYKEY123** for default codepage IBM1047.

```
STRUEN INFILE(MYLIB/QTXTSRC) INMBR(TESTLOGIN) OUTFILE(MYLIB/ENCRYPTEDF)
OUTMBR(ENCRYPTEDF) KEY(MYKEY123)
```

Figure 9.12 Universal Encrypt for OS/400 - Sample Command Syntax

The resulting encrypted command file is written to file **ENCRYPTEDF** in **MYLIB** library.

Figure 9.13, below, illustrates the contents of **MYLIB/ENCRYPTEDF(ENCRYPTEDF)**.

```
# Universal Encrypt
# Created on Wed Feb 22 18:43:51 2007
# Created by uencrypt 3.2.0 Level 0

9ACB96416816600CB9D24C9072D80C11768B93CB0E79B944EC37D3495097AD793F97399220C9BB
472DF1E04F5BA8909BCA6C8C72DFD3B706487B1713E6F73F5A0539F17076DEF6D14083EF6E7023
158526E70BE3AF688579805DCAC0CFF1EB6A
```

Figure 9.13 Universal Encrypt for OS/400 - Encrypted Command File

This encrypted file now can be used as command file input for Universal Product command on any platform that uses the encryption key **MYKEY123**.

9.5 Universal Encrypt for HP NonStop

This section describes Universal Encrypt configuration options and command line syntax for the HP NonStop operating system.

**Currently, HP NonStop runs Universal Encrypt 2.1.1.
This section provides information for that version.**

9.5.1 Configuration Options

Table 9.5, below, identifies the Universal Encrypt configuration options for HP NonStop.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
ENCRYPTION_KEY	Encryption key to use in the encryption algorithm
HELP	Writes a description of the command options and their format
VERSION	Writes the program version and copyright information

Table 9.5 Universal Encrypt for HP NonStop - Configuration Options

9.5.2 Command Line Syntax

Figure 9.14, below, illustrates the command line syntax – using the long form of configuration options – of Universal Encrypt for HP NonStop.

```
uencrypt
[-key key]

uencrypt
{ -help | -version }
```

Figure 9.14 Universal Encrypt for HP NonStop - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

9.5.3 Example

Assume that a command file named **cmdclear** contains the following data:

```
-u Thomas –w thames
```

Figure 9.15 Universal Encrypt for HP NonStop - Contents of Sample Command File

The following command encrypts the command file using an encryption key **MYKEY123**:

```
run uencrypt /IN cmdclear, OUT cmdenc/ –key MYKEY123
```

Figure 9.16 Universal Encrypt for HP NonStop - Sample Syntax.

The resulting encrypted command file is written to file **cmdenc**.

Figure 9.17, below, illustrates the contents of **cmdenc**.

```
# Universal Encrypt
# Created on Mon Jul 14 16:47:50 2003
# Created by uencrypt 2.1.1 Level 0

4F4813F7767318C3B1FB016F95B5FD07A6F90A787D9643A03C36503E761DF84AB64FF8877C76F9
8FDBEA1CE672A2DE943CE81BC1C159ABB01D0EC9E52E04A8C21A0269BE85F8443C1A5543901851
C29BE8223471A6BCD498163CD40D1E1866B4
```

Figure 9.17 Universal Encrypt for HP NonStop - Encrypted Command File

This encrypted command file now can be used by any Universal Product command, on any operating system. by specifying the encryption key **MYKEY123**.

Chapter 10
Universal Event Log Dump

10.1 Overview

Universal Event Log Dump (UELD) is a utility that selects records from one of the Windows event logs and writes them to a specified output file.

All records from a log can be dumped, or event records can be selected according to the date and time that they were generated.

UELD can be run any time as a stand-alone application. It also is designed to work with Universal Command, which provides centralized control from any operating system and additional options for redirecting output.

10.2 Usage

Universal Event Log Dump (UELD) consists of the command line program (**`ueld`**) followed by a list of configuration options. This section describes the command line input; that is the configuration options and their syntax on the command line.

10.2.1 Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UELD.
- Setting options and preferences for a single execution of UELD.

Configuration options are read from the following sources:

1. Command line
2. Configuration file

The order of precedence is the same as the list above; command line being the highest, and configuration file being the lowest. That is, options specified via a command line override options specified via the configuration file.

The configuration file, **`ueld.conf`**, provides the simplest method of specifying configuration options whose values will not change with each command invocation. These default values are used if the options are not read from one or more other sources.

Some options only can be specified in the configuration file; they have no corresponding command line equivalent. Other options cannot be specified in the configuration file; they must be specified via one or more other sources for a single execution of UCMD Manager.

See Section 2.2.1 Configuration Methods for details on configuration methods and command input for Universal Products.

10.2.2 Configuration Options

This section describes the configuration options used to execute UELD.

Configuration Options Categories

Table 10.1, below, categorizes the options into logical areas of application.

Category	Description
Local	Options required for local broker registration.
Log	Event records to select from which event log and what actions should be taken.
Messages	Utility message options.
Miscellaneous	Options use to display command help and program versions.
Output	How the event log records are printed.

Table 10.1 Universal Event Log Dump - Configuration Options Categories

The Universal Event Log Dump options for each category are summarized in the following tables.

Each **Option Name** is a link to detailed information about that option in the Universal Products Utilities 3.2.0 Reference Guide.

Local Category Options

Option Name	Description
INSTALLATION_DIRECTORY	Base directory where product is installed.
NLS_DIRECTORY	UMC and UTT file directory.

Log Category Options

Option Name	Description
BACKUP_LOG	Causes the log to be backed up before it is cleared.
CLEAR_ LOG	Causes the records in the log to be deleted from the log.
END_TIME	Ending date and time.
LOG_TYPE	Event log to be dumped.
REMOTE_SERVER	Name of a remote computer from which event log records should be retrieved.
START_TIME	Starting date and time.

Message Category Options

Option Name	Description
LOG_DIRECTORY	Log file directory.
MESSAGE_DESTINATION	Location where messages are written.
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file that will be used to write messages.
MESSAGE_LEVEL	Level of messages that will be displayed.

Miscellaneous Category Options

Option Name	Description
HELP	Writes a description of the command options and their format.
VERSION	Writes the program version and copyright information.

Output Category Options

Option Name	Description
OUTPUT_FILE	Complete path to the file that will be used to store the selected event log records.
PAGE_HEADERS	Enables or disables the printing of page headers.
PAGE_LENGTH	Number of lines that should be printed on each page.

10.2.3 Command Line Syntax

Figure 10.1 illustrates the command line syntax – using the long form of command line options – of UELD.

```
ueld
[-logtype {system|application|security}]
[-clear [-backup filename] ]
[-stime startdate [,starttime] ]
[-etime enddate [,endtime] ]
[-server servername]
[-file filename]
[-header {yes|no}]
[-length pagelength]
[-level {trace|audit|info|warn|error}]
[-dest {stderr|logfile}]
[-lang language]

ueld
{ -help | -version }
```

Figure 10.1 Universal Event Log Dump - Command Line Syntax

Configuration consists of setting default options and preferences. This section describes

10.3 Security

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.

10.3.1 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.

10.3.2 Universal Configuration Manager

To set configuration options for UELD using the Universal Configuration Manager, you must be logged on using an account that is a member of the Administrators group.

10.4 Examples of Universal Event Log Dump

This section contains examples demonstrating the use of Universal Event Log Dump (UELD).

The following List provides a link to each example:

- Execute Universal Event Log Dump from z/OS Manager
- Execute Universal Event Log Dump from a Windows Server

10.4.1 Execute Universal Event Log Dump from z/OS Manager

Figure 10.2, below, illustrates the execution of Universal Event Log Dump from a z/OS Universal Command Manager.

The application log, from the previous day at 15:00 until current time, will be dumped to the stdout of the manager process to be archived.

```
//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.userid(userid)
//SCRIPTDD DD *
ueld -t APPLICATION -s "*-1,15:00 PM"
//SYSIN    DD *
 -s SCRIPTDD
 -x LOGONDD
 -host dallas
/*
```

Figure 10.2 Universal Event Log Dump - Execution from z/OS Manager

The JCL procedure **UCMDPRC** is used to execute the **ueld** command. The command is sent to a remote system named **dallas** for execution. The **UNVOUT** DD in the **UCMDPRC** points to sysout, and is where the stdout of the remote command will be written. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**.

Command Line Options

The command line options used in this example are:

Command Options	Description
-t	Specifies which event log should be dumped.
-s	Specifies the starting date and time.

10.4.2 Execute Universal Event Log Dump from a Windows Server

Figure 10.3, below, illustrates the execution of Universal Event Log Dump from a Windows server.

The application log, from the previous day at 15:00 until current time, will be dumped to a file on the server.

```
ueld -t APPLICATION -s "*-1,15:00 PM" -f c:\application.log
```

Figure 10.3 Universal Event Log Dump - Execution from Windows Server

Command Line Options

The command line options used in this example are:

Command Options	Description
-t	Specifies which event log should be dumped.
-s	Specifies the starting date and time.
-f	Specifies the complete path to the file that will be used to store the selected event log records.

Chapter 11
Universal Message Translator

11.1 Overview

Universal Message Translator (UMET) translates error messages into return (exit) codes based on a user-defined translation table.

Every command ends with a return code that indicates the success or failure of the command execution. Typically, a return code of 0 indicates success; all other codes indicate failure.

However, a small number of commands do not set their return code under failure conditions; instead, they issue error messages. Based on the user-defined translation table, Universal Message Translator translates these error messages into return codes.

11.2 Usage

UMET requires two input files:

1. Message Input file (user-specified or standard input) containing the error messages that are to be translated into a return codes.
2. Translation Table file containing the user-defined translation table that controls the error message-to-return code translation process.

To perform a translation, UMET:

1. Reads the messages in the input file.
2. Matches each line against the translation table entries.
3. Exits with an return code from the best match in the translation table.

If no match is found, UMET ends with return code 0.

UMET performs operations specified by the configuration options. This section describes each option and their syntax.

11.2.1 Translation Table

The translation table specifies:

- Text to search for.
- Return code associated with the text.
- Precedence when multiple matches are found.

Translation Table Format

The translation table consists of one or more lines.

Each line is either:

- Comment line (# in column one)
- Blank line (ignored)
- Translation table entry

Translation table entries consist of two fields separated by spaces or tabs. An entry cannot be continued onto multiple lines.

Translation Table Fields

The translation table entry fields are:

Field	Description
Message Mask	Selects which messages to match in the input file. The mask must be enclosed in double (**"**) quotation marks. Mask characters include the asterisks (*****) and the question mark (**?**). The asterisk matches 0 or more characters and the question mark matches one character. If an asterisk, question mark, or quotation mark is required in the message text, it must be preceded with a back slash (****). If a back slash is required in the message text, it must be preceded by another back slash.
Exit Code	Specifies an integer value that UMET exits with if this entry is the resulting match. The exit code is in the range of –99999 to 99999.

Figure 11.1 Universal Message Translator – Translation Table

11.2.2 Matching Algorithm

The input file is read line by line. For each line, the line is compared to each entry in the translation table. All the matching entries are saved.

After the entire input file is read, the matched entries from the translation table are sorted in ascending order by their line number in the translation table. The first entry in this sorted list is the resulting translation table entry. The exit code from the resulting translation table entry is used as the return code of UMET. If no matching entry is found, UMET exits with 0.

11.3 Universal Message Translator for z/OS

This section describes the Universal Message Translator (UMET) for the z/OS operating system.

11.3.1 JCL

Figure 11.2, below, illustrates the Universal Message Translator for z/OS JCL.

```
//UMET     EXEC PGM=UMET,PARM='-TABLE TABLE'
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//TABLE    DD  DISP=SHR,MY.TRANS.TABLE
//SYSIN    DD  DISP=SHR,MY.MSG.FILE
```

Figure 11.2 Universal Message Translator for z/OS – JCL

UMET options are passed in with the PARM keyword on the EXEC statement.

11.3.2 DD Statements used in JCL Procedure

Table 11.1, below, describes the DD statements used in the Universal Message Translator for z/OS JCL illustrated in Figure 11.2.

ddname	Description
STEPLIB	Load library in which program UMET is located.
SYSPRINT	UMET standard output ddname.
SYSOUT	UMET standard error ddname.
TABLE	Translation table specified by the –table option on the PARM keyword.
SYSIN	Standard input ddname from which the message file is read.

Table 11.1 Universal Message Translator for z/OS – DD Statements in JCL

11.3.3 Configuration Options

Table 11.2 identifies the UMET for z/OS configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
HELP	Writes a description of the configuration options and their format.
IGNORE_CASE	Specification that matching of message masks to the input file is not case sensitive.
MESSAGE_FILE	Input message file name.
MESSAGE_LEVEL	Level of messages that will be written.
TRANSLATION_TABLE	Translation table file name.
VERSION	Writes the program version and copyright information.

Table 11.2 Universal Message Translator for z/OS - Configuration Options

11.3.4 Command Line Syntax

Figure 11.3, below, illustrates the syntax – using the long form of command line options – of UMET for z/OS.

```
umet
-table table
[-file messages]
[-ignorecase]
[-level {verbose|info|warn|error}]

umet
{ -help | -version }
```

Figure 11.3 Universal Message Translator for z/OS - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

11.4 Universal Message Translator for Windows and UNIX

This section describes Universal Message Translator (UMET) for the Windows and UNIX operating systems.

11.4.1 Configuration Options

Table 11.3, below, identifies the UMET for Windows and UNIX configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
HELP	Writes a description of the command options and their format.
IGNORE_CASE	Specification that matching of message masks to the input file is not case sensitive.
MESSAGE_FILE	Input message file name.
MESSAGE_LEVEL	Level of messages that will be written.
TRANSLATION_TABLE	Translation table file name.
VERSION	Writes the program version and copyright information.

Table 11.3 Universal Message Translator for Windows and UNIX - Configuration Options

11.4.2 Command Line Syntax

Figure 11.4, below, illustrates the syntax — using the long form of command line options — of UMET for Windows and UNIX.

```
umet
-table table
[-file messages]
[-ignorecase]
[-level {verbose|info|warn|error}]

umet
{ -help | -version }
```

Figure 11.4 Universal Message Translator for Windows and UNIX - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

11.5 Universal Message Translator for OS/400

This section describes Universal Message Translator (UMET) program for the OS/400 operating system.

11.5.1 Return Codes

The resulting return code from the translation process is converted into an OS/400 escape message.

The escape message ID and message severity depend on the return code value. as identified in Table 11.4, below.

Return Code	Message ID	Message Severity
1 – 10	UNV0344	10
11 – 20	UNV0345	20
21 – 30	UNV0346	30
31 and higher	UNV0347	40

Table 11.4 Universal Message Translator for OS/400 - Return Codes

11.5.2 Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This section references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Message Translator, substitute the tagged names for the untagged names in these references.

11.5.3 Configuration Options

Table 11.5, below, identifies the UMET for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
IGNORE_CASE	Specification that matching of message masks to the input file is not case sensitive.
MESSAGE_FILE	Input message file name.
MESSAGE_LEVEL	Level of messages that will be written.
TRANSLATION_TABLE	Translation table file name.

Table 11.5 Universal Message Translator for OS/400 - Configuration Options

11.5.4 Command Line Syntax

Figure 11.5 illustrates the syntax — using the STRUME parameter form of command line options — of UMET for OS/400.

```
STRUME
TBL([library/]{umetbl|filename}) [TBLMBR(member)]
[MSGFILE([library/]{*stdin|filename}) [MSGMBR(member)] ]

**Additional Options**
[MSGLEVEL(*{verbose|info|warn|error})]
[IGNORECASE({yes|no})]
```

Figure 11.5 Universal Message Translator for OS/400 - Command Line Syntax

11.6 Universal Message Translator for HP NonStop

This section describes Universal Message Translator (UMET) for the HP NonStop operating system.

11.6.1 Configuration Options

Table 11.6, below, identifies the UMET for HP NonStop configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
HELP	Writes a description of the command options and their format.
IGNORE_CASE	Specification that matching of message masks to the input file is not case sensitive.
MESSAGE_FILE	Input message file name.
MESSAGE_LEVEL	Level of messages that will be written.
TRANSLATION_TABLE	Translation table file name.
VERSION	Writes the program version and copyright information.

Table 11.6 Universal Message Translator for HP NonStop - Configuration Options

11.6.2 Command Line Syntax

Figure 11.6, below, illustrates the syntax – using the long form of command line options – of UMET for HP NonStop.

```
umet
-table table
[-file messages]
[-ignorecase]
[-level {verbose|info|warn|error}]

umet
{ -help | -version }
```

Figure 11.6 Universal Message Translator for HP NonStop - Command Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

11.7 Examples of Universal Message Translator

This section contains examples demonstrating the use of Universal Message Translator.

The following List provides a link to each example.

- Universal Message Translator: Example 1
- Universal Message Translator: Example 2
- z/OS: Execute Universal Message Translator from z/OS
- z/OS: Execute Universal Message Translator from z/OS Manager (in a Script with Table Housed on Remote Server)
- z/OS: Execute Universal Message Translator from z/OS Manager (in a Script with Table Housed on z/OS)
- Windows: Execute Universal Message Translator from Windows
- UNIX: Execute Universal Message Translator from UNIX
- OS/400: Execute Universal Message Translator from OS/400
- HP NonStop: Execute Universal Message Translator from HP NonStop

The OS/400 examples in this section reference the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Message Translator, substitute the tagged names for the untagged names. (See Section 11.5.2 Universal Products for OS/400 Commands for further information.)

11.7.1 Universal Message Translator: Example 1

Note: This example is not specific to a particular operating system.

In this example, a command generates the following **stderr** file:

```
Error opening rc file /etc/arc.rc
No rc file opened.
Ending due to error.
```

Figure 11.7 Universal Message Translator - Example 1, Message File

From the contents of the message file, we can see that the program failed to open a resource configuration file.

Either of the following translation tables could match error messages in the message file. Message masks should be general enough to match a set of error messages.

```
# UMET Translation Table 1
#
# Message Mask                Exit Code
# --------------------------- ------------
  “*error*”                   8
```

Figure 11.8 Universal Message Translator - Example 1, Translation Table 1

Translation Table 1 will result in a match if any input line contains the word **error**. The resulting exit code will be *8* if a match occurs.

```
# UMET Translation Table 2
#
# Message Mask                Exit Code
# --------------------------- ------------
  “Ending due to error.”      8
```

Figure 11.9 Universal Message Translator - Example 1, Translation Table 2

Translation Table 2 will result in a match only if the exact message text **"Ending due to error.**” appears as a line in the input file. This is less general, but may be sufficient for this command.

11.7.2 Universal Message Translator: Example 2

(This example continues from Universal Message Translator: Example 1.)

In this example, the command now generates the following **stderr** file:

```
Error opening rc file /etc/arc.rc
Processing rc file /usr/etc/arc.rc
Ending successfully
```

Figure 11.10 Universal Message Translator - Example 2, Message File

From the contents of the message file, we can see that the program failed to open a resource configuration file **/etc/arc.rc**, but successfully opened file **/usr/etc/arc.rc**.

The following translation table is one of many that could match error messages in the message file.

```
# UMET Translation Table 1
#
# Message Mask                Exit Code
# --------------------------- ------------
  “Ending due to error.”      8
  “Processing rc file *”      0
  “Error opening rc file *”    8
```

Figure 11.11 Universal Message Translator - Example 2, Translation Table 1

Translation Table 1 contains three entries:

- First entry matches against a specific error message that always indicates an error if present.
- Second and third entries match messages produced by resource configuration file processing.

11.7.3 z/OS: Execute Universal Message Translator from z/OS

Figure 11.12, below, illustrates the execution of Universal Message Translator from z/OS.

```
//stepname EXEC PGM=UMET,PARM='-TABLE TABLEDD -LEVEL VERBOSE'
//STEPLIB  DD  DISP=SHR,DSN=hlq.UNV.SUNVLOAD
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//TABLEDD  DD *
  "*ERROR*"       8
  "*WARN*"        4
  "*ERROR*"       7
/*
//SYSIN    DD *
THIS IS AN ERROR MESSAGE RESULTING IN RETURN CODE 8.
/*
```

Figure 11.12 Universal Message Translator - Execute from z/OS

The parameter **–TABLE** points to the DD statement **TABLEDD**, which defines the return codes to end this process based on matching text. The first column defines the text to match; the second defines the return code to set if the matching text exists in the **SYSIN** DD. The **–LEVEL** turns on messaging. All messages will be written to **SYSPRINT**. The **SYSIN** DD statement points to the text file to be interrogated.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-TABLE	Specifies the translation table file name.
-LEVEL	Specifies the level of messages that will be displayed.

11.7.4 z/OS: Execute Universal Message Translator from z/OS Manager (in a Script with Table Housed on Remote Server)

Figure 11.13, below, illustrates the execution of Universal Message Translator from a z/OS Universal Command Manager.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
/opt/universal/ucmdsrv-2.2.0/bin/umet -f /home/log.file -table\
 /home/umet.table -level verbose
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -userid joe
 -pwd abcdefg
/*
```

Figure 11.13 Universal Message Translator - Execute from z/OS Manager (with Table on Remote Server)

Universal Message Translator is executed in order to interrogate a log file and set the return code based on the translation table.

Since the command spans two lines, the native operating system continuation character must be used:

- / for UNIX
- ¬ for Windows

The full path to the Universal Message Translator executable must be specified for UNIX if the path is not part of the user's profile.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-table	Specifies the translation table file name.
-level	Specifies the level of messages that will be displayed.
-f	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

11.7.5 z/OS: Execute Universal Message Translator from z/OS Manager (in a Script with Table Housed on z/OS)

Figure 11.14, below, illustrates the execution of Universal Message Translator from a z/OS Universal Command Manager.

The message table is stored and maintained on z/OS and copied down to the server upon execution. The parameter **-t** points to the table of defined return codes based on text. The **-f** points to the text file to be interrogated.

```
//stepname EXEC UCMDPRC
//UNVIN    DD DISP=SHR,DSN=hlq.umet.table
//SCRIPTDD DD *
UCOPY > c:\temp\umet.table
umet -t c:\temp\umet.table -f c:\temp\bkup.log -level verbose
/*
//SYSIN   DD  *
 -host dallas
 -script SCRIPTDD
 -userid joe
 -pwd abcdefg

/*
```

Figure 11.14 Universal Message Translator - Execute from z/OS Manager (with Table on z/OS)

The first command copies the messages table from the **UNVIN** DD of the manager process to a server file named **c:\temp\umet.table**. The UMET program is then executed to interrogate the log file and set the return code based on the translation table.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-table	Specifies the translation table file name.
-level	Specifies the level of messages that will be displayed.
-f	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

11.7.6 Windows: Execute Universal Message Translator from Windows

Figure 11.15, below, illustrates the execution of Universal Message Translator from Windows.

```
umet -t c:\umettable.txt -f c:\umetfile.txt -level verbose
```

Figure 11.15 Universal Message Translator - Execute from Windows

The parameter **-t** points to the file that defines the return codes with which to end this process, based on matching text. The first column of the file defines the text to match; the second defines the exit code to set if the matching text exists in the file defined by **-f**. The **-level** option turns on messaging. All messages will be written to **stdout**.

Command Line Options

The command line options used in this example are:

Option	Description
-t	Specifies the translation table file name.
-level	Specifies the level of messages that will be displayed.
-f	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

11.7.7 UNIX: Execute Universal Message Translator from UNIX

Figure 11.16, below, illustrates the execution of Universal Message Translator from UNIX.

Although the command is shown on two lines, it should be entered on one line at the command prompt or within a script, or it can be continued within the script with the UNIX continuation character \.

```
/opt/universal/ucmdsrv-2.2.0/bin/umet -t /tmp/umettable.txt -f
/tmp/umetfile.txt -level verbose
```

Figure 11.16 Universal Message Translator - Execute from UNIX

The parameter **-t** points to the file, which defines the return codes with which to end this process, based on matching text. The first column of the file defines the text to match; the second defines the return code to set if the matching text exists in the file defined by **-f**. All messages will be written to **stdout**.

Command Line Options

The command line options used in this example are:

Option	Description
-t	Specifies the translation table file name.
-level	Specifies the level of messages that will be displayed.
-f	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

11.7.8 OS/400: Execute Universal Message Translator from OS/400

Figure 11.17, below, illustrates the execution of Universal Message Translator from OS/400.

```
STRUME MSGFILE(input_file) MSGMBR(member) TBL(table_file) TBLMBR(member)
MSGLEVEL(*VERBOSE)
```

Figure 11.17 Universal Message Translator - Execute from OS/400

The parameter **TBL/TBLMBR** points to the file, which defines the exit codes with which to end this process, based on matching text. The first column of the file defines the text to match; the second defines the return code to set if the matching text exists in the file defined by **MSGFILE/MSGMBR**. All messages will be written to **stdout**.

Command Line Options

The command line options used in this example are:

Option	Description
-TBL [TBLMBR]	Specifies the translation table file name.
-MSGLEVEL	Specifies the level of messages that will be displayed.
-MSGFILE [MSGMBR]	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

11.7.9 HP NonStop: Execute Universal Message Translator from HP NonStop

Figure 11.18, below, illustrates the execution of Universal Message Translator from HP NonStop.

```
run $SYSTEM.UNVBIN.umet -table umettable -file umetfile -level verbose
```

Figure 11.18 Universal Message Translator - Execute from OS/400

The parameter **-t** points to the file, which defines the exit codes with which to end this process, based on matching text. The first column of the file defines the text to match; the second defines the return code to set if the matching text exists in the file defined by **-f**. All messages will be written to stdout.

Command Line Options

The command line options used in this example are:

Option	Description
-table	Specifies the translation table file name.
-level	Specifies the level of messages that will be displayed.
-file	Specifies the input message file name. If the option is not specified, UMET reads its input from **stdin**.

Chapter 12
Universal Products Install Merge

12.1 Overview

The Universal Products Install Merge (UPIMERGE) utility merges options and values from one Universal Products configuration file or component definition file with another.

UPIMERGE runs automatically during Universal Products installation upgrades on UNIX and Windows. During the install, UPIMERGE combines options and values from existing configuration and component definition files with the options and values in the most recent versions of those files (delivered with the distribution package).

The result of each merge is a single file, with preserved options and values residing alongside any new options and values that were introduced to support new Universal Product features.

The Universal Products (UNIX and Windows) and Universal Enterprise Controller (Windows only) distribution packages also install UPIMERGE. This makes UPIMERGE available at any time for recovering archived options and values and merging them with the most recent options and values.

When used to update a Universal Products configuration or component definition file, UPIMERGE must run with a user account that has write access to the output file. This typically means administrative access (that is, root on UNIX, Administrator on Windows).

12.2 Usage

As input, UPIMERGE typically uses an archived configuration file or component definition file. However, it can use any file as input, provided that the file is in standard keyword / value format.

UPIMERGE output is a file containing the options and values from the input file, merged with those in the output file. For each option in the output file, UPIMERGE replaces its value with the value of a matching option in the input file. If the input file contains options not defined in the output file, UPIMERGE adds those options to the end of the output file.

UPIMERGE does not attempt to sequence the entries that it adds to the output file. Thus, the order of options in the output file may not match the order of the same options in the input file.

If the input file contains more than one entry for an option, UPIMERGE adds every entry to the output file. The application will use the value of the last entry that appears in the output file.

UPIMERGE does not update any comments in the output file. Options that reside only in the output file are either commented out or left as is, depending on the command line parameters specified.

Note: UPIMERGE has no effect on a Universal Products application's behavior if the local Universal Broker is in managed mode. In that environment, configurations and component definitions reside in a database file, not a text file. Use the Universal Management Console application to manage configurations for managed installations.

12.2.1 Configuration Options

Table 12.1, below, identifies the Universal Products Install Merge configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
BACKUP_DESTINATION	Creates a copy of the original DESTINATION_FILE prior to the merge.
COMMAND_FILE_ENCRYPTED	Name of a file encrypted with Universal Encrypt that contains command options.
COMMAND_FILE_PLAIN	Name of a plain text file that contains command options.
COMPONENT_TYPE	Notifies UPIMERGE that the SOURCE_FILE is a component definition file that contains settings for the specified Universal Products server component. You cannot use this option with CONFIGURATION_TYPE. UPIMERGE ignores this option if INSTALLATION_DIRECTORY is omitted.
CONFIGURATION_TYPE	Notifies UPIMERGE that the SOURCE_FILE is a configuration file that contains settings for the specified Universal Products application. You cannot use this option with COMPONENT_TYPE. UPIMERGE ignores this option if INSTALLATION_DIRECTORY is omitted.
DESTINATION_FILE	Name of a file used to store the result of the merge.
ENCRYPTION_KEY	Key used to encrypt the file specified by COMMAND_FILE_ENCRYPTED.
HELP	Writes a description of the command options and their format.
INSTALLATION_DIRECTORY	Primary location in which the Universal Products server component identified by COMPONENT_TYPE, or the Universal Products application identified by CONFIGURATION_TYPE, resides.
KEEP_NOMATCH	Controls merge behavior when an option in DESTINATION_FILE has no match in SOURCE_FILE.
MESSAGE_LEVEL	Level of messages to write.
SOURCE_FILE	Name of a file used as input to the merge. If this parameter is omitted, UPIMERGE assumes input is redirected via stdin.
VERSION	Writes program version and copyright information.

Table 12.1 Universal Products Install Merge - Configuration Options

12.2.2 Command Line Syntax

Figure 12.1, below, illustrates the syntax – using the long form of command line options – of Universal Products Install Merge.

```
upimerge
-dest filename
[ -source filename ]
[ -installdir dirname { -cfgtype type [-comptype type] | -comptype type
   [-cfgtype type] } ]
[ -keep_nomatch {yes | no} ]
[ -bkup_dest {yes | no} ]
[ -file filename | -encryptedfile filename [-key key] ]
[ -level { trace | audit | info | warn | error }[,{ time | notime }]  ]

upimerge
{ -help | -version }
```

Figure 12.1 Universal Products Install Merge - Command Line Syntax

12.3 Examples of Universal Products Install Merge

This section contains examples demonstrating the use of Universal Products Install Merge.

The following list provides a link to each example.

- Merge Files Using Program Defaults
- Merge Files Introducing New Options
- Merge Files Using Installation-Dependent Values

The examples in this section demonstrate the expected results when UPIMERGE is executed using two files with the contents identified in Table 12.2 and Table 12.3 (see Files Used in Examples).

Note: Although these examples show Windows path names, the UPIMERGE behavior demonstrated also applies to UNIX systems.

Files Used in Examples

Table 12.2, below, identifies the contents of **infile.txt**, a sample file in Universal Products' standard keyword / value configuration file format.

For the examples in this section, **infile.txt** could represent an existing or archived configuration file, or a work file used to introduce and distribute configuration values across one or more target systems.

Keyword	Value
installation_directory	“C:\Program Files\Universal\UCmdMgr”
message_level	info
#host	some.remote.host
port	7850
license_product	"UNIVERSAL COMMAND MANAGER"
license_customer	"STONEBRANCH, INC."
license_type	DEMO
license_expiration_date	2012.12.21
license_nt_servers	1
license_key	078B-E180-64E6-3016-EA20-0CF4-58F9-B301 *
* This license key is for demonstration purposes only. It is not a valid license key.	

Table 12.2 Universal Products Configuration File Sample (infile.txt)

Table 12.3, below, identifies the contents of **outfile.txt**, another sample file in Universal Products’ standard keyword / value configuration file format.

For the examples in this section, **outfile.txt** might represent a default configuration file that is delivered during product installation, or an existing production configuration file that needs to be updated with values from **infile.txt**.

Keyword	Value
port	7887
activity_monitoring	yes
event_generation	*,x100

Table 12.3 Universal Products Configuration File Sample (outfile.txt)

12.3.1 Merge Files Using Program Defaults

Figure 12.2, below, illustrates the command line used to merge configuration options from **infile.txt** into **outfile.txt**.

In this example, UPIMERGE executes using program defaults.

```
upimerge -dest outfile.txt -source infile.txt
```

Figure 12.2 Merge infile.txt into outfile.txt using program defaults

Table 12.4, below, identifies the contents of **outfile.txt** after UPIMERGE completes.

To obtain this result, UPIMERGE added options from **infile.txt** that did not exist in **outfile.txt** (that is, installation_directory, message_level, license_key, and so on). It also preserved the value for the **port** option by replacing the 7887 value with the currently defined 7850.

UPIMERGE also dropped the commented **host** option from **infile.txt**. UPIMERGE ignores any comments in the input file, because merging those lines into the output file would have no effect on the application's behavior.

Finally, UPIMERGE commented out the **activity_monitoring** and **event_generation** options introduced by **outfile.txt**. UPIMERGE cannot distinguish between options for new features and new values for existing options. To prevent the introduction of a new value into an application currently running with application-defined defaults, UPIMERGE's default response is to comment out any option in the output file with no match in the input file.

Keyword	Value
installation_directory	"C:\Program Files\Universal\UCmdMgr"
message_level	info
port	7850
license_product	"UNIVERSAL COMMAND MANAGER"
license_customer	"STONEBRANCH, INC."
license_type	DEMO
license_expiration_date	2012.12.21
license_nt_servers	1
license_key	078B-E180-64E6-3016-EA20-0CF4-58F9-B301
#activity_monitoring	yes
#event_generation	*,x100

Table 12.4 Contents of outfile.txt after default merge

12.3.2 Merge Files Introducing New Options

Figure 12.3, below, illustrates the command line used to merge configuration options from **infile.txt** into **outfile.txt**.

In this example, UPIMERGE changes its default behavior, and introduces new values for the **activity_monitoring** and **event_generation** options by not commenting them out in the merged file.

```
upimerge -dest outfile.txt -source infile.txt
-keep_nomatch yes
```

Figure 12.3 Merge infile.txt into outfile.txt keeping new options

Table 12.5, below, identifies the contents of **outfile.txt** after UPIMERGE completes.

The result is almost identical to the example shown in Table 12.4. Executing UPIMERGE with **-keep_nomatch** set to **yes** enables the **activity_monitoring** and **event_generation** options in the output file.

Keyword	Value
installation_directory	“C:\Program Files\Universal\UCmdMgr”
message_level	info
port	7850
license_product	"UNIVERSAL COMMAND MANAGER"
license_customer	"STONEBRANCH, INC."
license_type	DEMO
license_expiration_date	2012.12.21
license_nt_servers	1
license_key	078B-E180-64E6-3016-EA20-0CF4-58F9-B301
activity_monitoring	yes
event_generation	*,x100

Table 12.5 Contents of outfile.txt when keeping unmatched destination values

12.3.3 Merge Files Using Installation-Dependent Values

Figure 12.4, below, illustrates the command line used to merge configuration options from **infile.txt** into **outfile.txt**.

In this example, UPIMERGE applies logic specific to a particular configuration file, and updates any references to locations that depend on the installed location of that Universal Product application.

```
upimerge -dest outfile.txt -source infile.txt
-cfgtype ucmd
-installdir "D:\Program Files\Universal\UCmdMgr"
```

Figure 12.4 Merge infile.txt into outfile.txt using installation-dependent values

Table 12.6, below, identifies the contents of **outfile.txt** after UPIMERGE completes. The result is almost identical to the example shown in Table 12.4, except for the value of the **-installdir** option.

Even though **infile.txt** contained a value for**-installdir**, UPIMERGE interpreted that value as the application's current location. UPIMERGE then updated any values in **outfile.txt** (executing logic based on the specified **–cfgtype**) that depend on the installed location.

This example might be useful in a situation where it is necessary to recover configuration settings from an archived file, but the application no longer resides in the directory specified in the archive file.

This is the logic that UPIMERGE uses during a Universal Products installation to ensure that installation-dependent locations are always correct.

Keyword	Value
installation_directory	“D:\Program Files\Universal\UCmdMgr”
message_level	info
Port	7850
license_product	"UNIVERSAL COMMAND MANAGER"
license_customer	"STONEBRANCH, INC."
license_type	DEMO
license_expiration_date	2012.12.21
license_nt_servers	1
license_key	078B-E180-64E6-3016-EA20-0CF4-58F9-B301
#activity_monitoring	yes
#event_generation	*,x100

Table 12.6 Contents of outfile.txt when using installation-dependent values

Chapter 13
Universal Query

13.1 Overview

The Universal Query (UQUERY) utility queries any Universal Broker for Broker-related and active component-related information.

13.1.1 Usage

UQUERY returns information for a Universal Broker that is installed on the host, as specified by configuration options on the command line or in a configuration file. Information regarding the components managed by a particular Broker also can be requested.

UQUERY registers with a locally running Universal Broker. Consequentially, a Universal Broker must be running in order for a UQUERY to execute.

13.2 Universal Query for z/OS

This section describes Universal Query (UQUERY) for the z/OS operating system.

13.2.1 JCL Procedure

Figure 13.1, below, illustrates the Universal Query for z/OS JCL procedure (**UQRYPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//UQRYPRC  PROC UPARM=,               -- UQUERY options
//             UCMDPRE=#SHLQ.UNV
//*
//PS1      EXEC PGM=UQUERY,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UCMDPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UCMDPRE..SUNVNLS
//UNVTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
```

Figure 13.1 Universal Query for z/OS – JCL Procedure

The parameter **UPARM** is used to specify EXEC PARM keyword values for Universal Query. The PARM values to the left of the slash (**/**) character are IBM Language Environment parameters.

(See the Universal Products 3.2.0 Installation Guide for information regarding the customization of Language Environment parameters.)

13.2.2 DD Statements used in JCL Procedure

Table 13.1, below, describes the DD statements used in the Universal Query for z/OS JCL procedure illustrated in Figure 13.1.

ddname	Description
STEPLIB	Load library in which program UQUERY is located.
UNVNLS	UQUERY national language support ddname.
UNVTRACE	UQUERY trace ddname.
SYSPRINT	UQUERY standard output ddname.
SYSOUT	UQUERY standard error ddname.

Table 13.1 Universal Query for z/OS – DD Statements in JCL Procedure

13.2.3 JCL

Figure 13.2, below, illustrates the Universal Query for z/OS JCL using the **UQRYPRC** procedure illustrated in Figure 13.1.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UQRYPRC
//SYSIN    DD  *
 -i dallas
/*
```

Figure 13.2 Universal Query for z/OS – JCL

Job step STEP1 executes the procedure **UQRYPRC**.

The command options are specified on the **SYSIN** DD.

13.2.4 Configuration Options

UQUERY for z/OS operations are controlled by the configuration options, which are specified either on the command lie (via the PARM keyword of the EXEC statement or in the SYSIN ddname) or the configuration file.

Table 13.2, below, identifies the UQUERY configuration options for z/OS.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CODE_PAGE	Character code page used to translate text data received and transmitted over the network.
COMMAND_ID	Requests that Universal Query return information for all records that match the specified command ID.
COMPONENT_ID	Requests that Universal Query return information only for the specified component ID.
HELP	Writes a description of the configuration options and their format.
MANAGERS	Specification for whether or not Universal Query requests manager component information from the queried Broker.
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file used to write messages.
MESSAGE_LEVEL	Level of messages to write.
OUTBOUND_IP	Sets the host or IP address that UQUERY binds to when initiating outgoing connections.
PING	Information Universal Query requests from Universal Broker.
REMOTE_HOST	IP address of the remote computer.
REMOTE_PORT	TCP port number on the remote computer on which Universal Broker is accepting connections.
REPORT	Format in which Universal Broker information is written.
SYSTEM_ID	Local Universal Broker with which the Universal Query must register.
VERSION	Writes the program version and copyright information.

Table 13.2 Universal Query for z/OS - Configuration Options

13.2.5 Command Line Syntax

Figure 13.3, below, illustrates the command line syntax — using the command line, long form of the configuration options — of UQUERY for z/OS.

```
uquery
-host hostaddress
[-port port]
[-system_id ID]
[-ping {yes|no}]
[-report {normal|fixed}]
[-codepage codepage]
[-level {trace|audit|info|warn|error}]
[-lang language]
[-managers {yes|no}]
[-outboundip host]
[-cmdid ID]
[-component ID]

uquery
{ -help | -version }
```

Figure 13.3 Universal Query for z/OS - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

13.3 Universal Query for UNIX and Windows

This section describes Universal Query (UQUERY) for the UNIX and Windows operating systems.

13.3.1 Configuration Options

Table 13.3, below, identifies the UQUERY configuration options for UNIX and Windows.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
BIF_DIRECTORY *	Broker Interface File (BIF) directory where the Universal Broker interface file is located.
CODE_PAGE	Character code page used to translate text data received and transmitted over the network.
COMMAND_ID	Requests that Universal Query return information for all records that match the specified command ID.
COMPONENT_ID	Requests that Universal Query return information only for the specified component ID.
HELP	Writes a description of the command options and their format.
MANAGERS	Specification for whether or not Universal Query requests manager component information from the queried Broker.
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file used to write messages.
MESSAGE_LEVEL	Level of messages to write.
NLS_ DIRECTORY	Directory where the Universal Query message catalog and code page tables are located.
OUTBOUND_IP	Sets the host or IP address that UQUERY binds to when initiating outgoing connections.
PING	Information Universal Query requests from Universal Broker.
PLF_DIRECTORY *	Program Lock File (PLF) directory where the program lock files are located.
REMOTE_HOST	IP address of the remote computer.
REMOTE_PORT	TCP port number on the remote computer on which Universal Broker is accepting connections.
REPORT	Format in which Universal Broker information is written.
VERSION	Writes the program version and copyright information.
* Valid for UNIX only.	

Table 13.3 Universal Query for UNIX and Windows - Configuration Options

13.3.2 Command Line Syntax

Figure 13.4, below, illustrates the command line syntax — using the command line, long form of the configuration options — of UQUERY for Windows and UNIX.

```
uquery
-host hostaddress
[-port port]
[-ping {yes|no}]
[-report {normal|fixed}]
[-bif_directory directory]    (NOTE: This option is valid only for UNIX.)
[-plf_directory directory]    (NOTE: This option is valid only for UNIX.)
[-codepage codepage]
[-level {trace|audit|info|warn|error}]
[-lang language]
[-managers {yes|no}]
[-outboundip host]
[-cmdid ID]
[-component ID]

uquery
{ -help | -version }
```

Figure 13.4 Universal Query for UNIX and Windows - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

Windows

The Universal Configuration Manager also can be used to obtain the same information reported by the command line version of Universal Query (see Section 2.4 Universal Configuration Manager).

13.4 Universal Query for OS/400

This section describes Universal Query (UQUERY) for the OS/400 operating system.

13.4.1 Universal Products for OS/400 Commands

The names of the Universal Products for OS/400 commands that are installed in the OS/400 **QSYS** library are tagged with the Universal Products for OS/400 **v**ersion / **r**elease / **m**odification number, **320**. The names of the commands installed in the Universal Products for OS/400 product library, **UNVPRD320**, are untagged.

To maintain consistency across releases, you may prefer to use the untagged names in your production environment. The **UCHGRLS** (Change Release Tag) program lets you change the tagged command names in **QSYS** to the untagged command names in **UNVPRD320**.

(See the Universal Products 3.2.0 for OS/400 Installation Guide for detailed information on **UCHGRLS**.)

This section references the OS/400 commands by their untagged names. If you are using commands with tagged names to run Universal Query, substitute the tagged names for the untagged names in these references.

13.4.2 Configuration Options

Table 13.5, below, identifies the UQUERY for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CODE_PAGE	Character code page used to translate text data received and transmitted over the network.
COMMAND_ID	Requests that Universal Query return information for all records that match the specified command ID.
COMPONENT_ID	Requests that Universal Query return information only for the specified component ID.
HELP	Writes a description of the configuration options and their format.
MANAGERS	Specification for whether or not Universal Query requests manager component information from the queried Broker.
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file used to write messages.
MESSAGE_LEVEL	Level of messages to write.
PING	Information Universal Query requests from Universal Broker.
PLF_DIRECTORY	Program Lock File (PLF) directory where the program lock files are located.
REMOTE_HOST	IP address of the remote computer.
REMOTE_PORT	TCP port number on the remote computer on which Universal Broker is accepting connections.
REPORT	Format in which Universal Broker information is written.
VERSION	Writes the program version and copyright information.

Table 13.4 Universal Query for OS/400 - Configuration Options

13.4.3 Command Line Syntax

Figure 13.5, below, illustrates the command line syntax — using the STRUQR parameter form of command line options — of UQUERY for OS/400.

```
STRUQR
HOST(hostaddress)
[PORT(port)]
[PING(*{yes|no})]
[REPORT(*{normal|fixed})]
[CODEPAGE(codepage)]
[MSGLANG(language)]
[MSGLEVEL(*{trace|audit|info|warn|error}]
          (NOTE: trace turns on the trace function.)
[OUTBOUNDIP(host|blank_line)]
[CMDID(ID)]
[COMPID(ID)]
[MANAGERS(*{yes|no})]
[PLFDIR(ifs_directory)]

STRUQR
VERSION(*{yes|no})
```

Figure 13.5 Universal Query for OS/400 - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

13.5 Universal Query for HP NonStop

This section describes Universal Query (UQUERY) for the HP NonStop operating system.

13.5.1 Configuration Options

Table 13.5, below, identifies the UQUERY for HP NonStop configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CODE_PAGE	Character code page used to translate text data received and transmitted over the network.
HELP	Writes a description of the configuration options and their format.
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file used to write messages.
MESSAGE_LEVEL	Level of messages to write.
PING	Information Universal Query requests from Universal Broker.
REMOTE_HOST	IP address of the remote computer.
REMOTE_PORT	TCP port number on the remote computer on which Universal Broker is accepting connections.
REPORT	Format in which Universal Broker information is written.
VERSION	Writes the program version and copyright information.

Table 13.5 Universal Query for HP NonStop - Configuration Options

13.5.2 Command Line Syntax

Figure 13.6, below, illustrates the command line syntax — using the command line, long form of the configuration options — of UQUERY for HP NonStop.

```
uquery
-host hostaddress
[-port port]
[-ping {yes|no}]
[-report {normal|fixed}]
[-codepage codepage]
[-level {trace|audit|info|warn|error}]
[-lang language]

uquery
{ -help | -version }
```

Figure 13.6 Universal Query for HP NonStop - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

13.6 Examples of Universal Query

This section contains examples demonstrating the use of Universal Query.

The following List provides a link to each example.

- Universal Query Output
- Universal Query for z/OS
- Universal Query for UNIX and Windows
- Universal Query for HP NonStop

13.6.1 Universal Query Output

Figure 13.7, below, illustrates an example of the output generated by the execution of the Universal Query command.

This sample output is from the execution of Universal Query to host **dallas.domain.com** using a NORMAL report.

```
                    Universal Query Report
                            for
                Mon 20 Jul 2009 05:54:00 PM EDT

        host: 10.20.30.40  port: 7887  ping: NO  report: NORMAL

               Ubroker Host Name...:
               Ubroker IP Address..: *
               Ubroker Host Port...: 7887
               Ubroker Description.: Universal Broker
               Ubroker Version.....: 3.2.0 Level 0 Release Build 108
               Ubroker Service.....: UNKNOWN
               Ubroker Status......: Active


         Component ID..............: 1121367481
         Component Name............: ucmd
         Component Description.....: Universal Command Server
         Component Version.........: 3.2.0 Level 0 Release Build 108
         Component Type............: ucmd
         Component Process ID......: 773
         Component Start Time......: 05:53:39 PM
         Component Start Date......: 07/20/2009
         Component Command ID......: sleep 60
         Component State...........: REGISTERED
         Component MGR UID.........: ucuser
         Component MGR Work ID.....: PID12890
         Component MGR Host Name...: dallas.domain.com
         Component MGR IP Address..: 10.20.30.34
         Component MGR Port........: 49082
         Component Comm State......: ESTABLISHED
         Component Comm State Time.: 05:53:41 PM
         Component Comm State Date.: 07/20/2009
         Component MGR Restartable.: NO
         Component Comment.........: Sleep for 60 secs on dallas
```

Figure 13.7 Universal Query Output

13.6.2 Universal Query for z/OS

The Universal Query is used to list all active components on a remote server.

The output will be written to the **SYSPRINT** DD statement.

```
//stepname EXEC UQRYPRC
//SYSIN    DD  *
-host dallas
/*
```

Figure 13.8 Universal Query for z/OS - JCL Procedure

All active component information for server dallas will be printed to DD statement **SYSOUT**.

SYSIN Option

The SYSIN option used in this example is:

Option	Description
-host	Directs the command to a computer with a host name of **dallas**.

13.6.3 Universal Query for UNIX and Windows

The Universal Query is used to list all active components on a remote server.

The output will be written to stdout.

```
uquery -host localhost
```

Figure 13.9 Universal Query for z/OS - JCL Procedure

All active component information for the **`localhost`** server will be printed to stdout.

Command Line Option

The command line option used in this example is:

Option	Description
`-host`	Directs the command to the **`localhost`**.

13.6.4 Universal Query for HP NonStop

The Universal Query is used to list all active components on a remote server.

The output will be written to stdout.

```
run $SYSTEM.UNVBIN.uquery -host localhost
```

Figure 13.10 Universal Query for z/OS - JCL Procedure

All active component information for the **localhost** server will be printed to stdout.

Command Line Option

The command line option used in this example is:

Option	Description
-host	Directs the command to the **localhost**.

Chapter 14
Universal Return Code

14.1 Overview

The Universal Return Code utility is a Windows utility that performs the function of ending a process with a return code that is equal to its command line argument.

The return code of a Windows batch script is the return code of the last command executed. Universal Return Code can be used as the last command to set the return code of the batch script.

14.2 Usage

The Universal Return Code program is **`urc.exe`**.

It exits with its integer command line argument as its return code.

14.2.1 Command Line Syntax

Figure 14.1, below, illustrates the syntax - using the command line, long form of the configuration options - of Universal Return Code.

```
urc
return_code

urc
{ -help | -version }
```

Figure 14.1 Universal Return Code – Command Line Syntax

14.2.2 Configuration Options

The command line arguments to Universal Return Code are:

Argument	Description
return_code	Integer-value return code of Universal Return Code.
-? | -h | -help	Write command instructions.
-v | -version	Write version information.

14.3 Examples of Universal Return Code

This section contains examples demonstrating the use of Universal Return Code.

The following list provides a link to each example.

- Universal Command Manager for z/OS executing Universal Return Code within a Script
- Universal Command Manager for z/OS executing Universal Return Code and Universal Message Translator within a Script

14.3.1 Universal Command Manager for z/OS executing Universal Return Code within a Script

Figure 14.2, below, illustrates the use of Universal Return Code to exit with the return code of a command in the middle of the script.

By default, the return code of the last command within the script sets the return code of the script. Universal Return Code is useful when multiple commands are executed within one script.

A user variable called RC is set to the value of the **ERRORLEVEL** of the previous command.

The last line of the script then uses that value as the URC value to set the return code of the script equal to the return code of the **backup.exe** program.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
backup.exe > c:\temp\bkup.log
SET RC=%ERRORLEVEL%
   UCOPY c:\temp\bkup.log
   DEL c:\temp\bkup.log
URC %RC%
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 14.2 Universal Return Code - Universal Command Manager for z/OS Executing URC within a Script

The first command executes a backup script.

The next step sets a variable called RC to the value of the return code of **backup.exe**.

The **UCOPY** command copies the log file to the Universal Command Manager.

The next step deletes the log file.

The last line of the script then uses the variable **RC** as the URC value in order to set the return code of the script equal to the exit code of the **backup.exe** execution, instead of the return code of the **DEL** command.

14.3.2 Universal Command Manager for z/OS executing Universal Return Code and Universal Message Translator within a Script

Figure 14.3, below, provides an example that builds onto the Figure 14.2 example by adding a step that executes the Universal Message Translator (UMET) utility.

UMET could be used if the first command does not set the return code properly. The example exits with the return code of a command in the middle of the script with the use of Universal Return Code. A user variable called RC is set to the value of the return code of the UMET execution. The last line of the script then uses that value as the URC value to set the return code of the script equal to the exit code of the UMET execution.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
backup.exe > c:\temp\bkup.log
umet -t c:\temp\translate.table -f c:\temp\bkup.log
SET RC=%ERRORLEVEL%
   UCOPY c:\temp\bkup.log
   DEL c:\temp\bkup.log
URC %RC%
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 14.3 Universal Return Code - Universal Command Manager for z/OS Executing URC and UMET within a Script

The first command executes a backup script.

The second command executes the UMET program and sets the return code of UMET based on the table definitions and the file being interrogated.

The next step sets a variable called RC to the value of the return code of the UMET execution.

The UCOPY command copies the log file to the Universal Command Manager.

The next step deletes the log file.

The last line of the script then uses the variable RC as the URC value in order to set the return code of the script equal to the return code of the UMET execution instead of the return code of the DEL command.

Chapter 15
Universal Spool List

15.1 Overview

The Universal Spool List (**USLIST**) utility provides the ability to list Universal Spool database records. Universal Spool List must be executed on the system on which the database is located.

The functions that Universal Spool List provide are required for possible database clean-up or problem resolution by Stonebranch, Inc. Customer Support.

15.1.1 Usage

The Universal Spool List utility reads requested records from a specified database. The selected records are written to standard output.

Universal Spool List performs operations specified by the command options.

15.1.2 Security

The account used to execute the Universal Spool List utility must have read access to the database files listed in Section 15.2.1 Databases.

15.2 Universal Spool List for z/OS

This section describes Universal Spool List for the z/OS operating system.

15.2.1 Databases

Universal Spool databases are implemented as HFS data sets. The HFS data sets **UNVDB** and **UNVSPOOL** contain an HFS file system that contains the Universal Spool database files.

Table 15.1, below, identifies the database files and the HFS data sets in which they reside.

Database Name	Data Set	File Name
Universal Broker Component Database	UNVDB	bcomponent.db
Universal Server Component Database	UNVDB	scomponent.db
Universal Server Spool Databases	UNVSPOOL	spool.stdin.*COMPID*.db spool.stdout.*COMPID*.db spool.stderr.*COMPID*.db
Note: In the Universal Server Spool Databases file names, *COMPID* is the component ID assigned to the Server instance.		

Table 15.1 Universal Spool – Databases

15.2.2 JCL Procedure

Figure 15.1, below, illustrates the Universal Spool List for z/OS JCL procedure (**USLLSPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//USLLSPRC PROC UPARM=,               -- USLIST options
//             UNVPRE=#SHLQ.UNV,
//             CFGPRE=#PHLQ.UNV,
//             DBPRE=#PHLQ.UNV
//*
//PS1      EXEC PGM=USLIST,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UNVPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UNVPRE..SUNVNLS
//UNVCONF  DD  DISP=SHR,DSN=&CFGPRE..UNVCONF
//UNVDB    DD  DISP=SHR,DSN=&DBPRE..UNVDB
//UNVSPOOL DD  DISP=SHR,DSN=&DBPRE..UNVSPOOL
//UNVTRACE DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

If zFS data sets are being used instead of the default HFS data sets, the UNVDB and UNVSPOOL ddnames cannot be used to allocate the zFS data sets. The ddnames must be removed along with the DBPRE procedure parameter.

Figure 15.1 Universal Spool List for z/OS – JCL Procedure

15.2.3 DD Statements used in JCL Procedure

Table 15.2, below, describes the DD statements used in the Universal Spool List for z/OS JCL procedure illustrated in Figure 15.2.

ddname	Description
STEPLIB	Load library in which program USLIST program is located.
UNVNLS	Universal National Language Support library.
UNVCONF	Universal Products configuration library.
UNVDB	Universal Broker Database HFS data set.
UNVSPOOL	Universal Spool Database HFS data set.
UNVTRACE	Application trace ddname.
SYSOUT	USLIST standard error ddname.
SYSPRINT	USLIST standard output ddname.

Table 15.2 Universal Spool List for z/OS – DD Statements in JCL Procedure

15.2.4 JCL

Figure 15.2, below, illustrates the Universal Spool List for z/OS JCL.

```
//STEP1    EXEC PGM=USLIST
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//UNVNLS   DD  DISP=SHR,DSN=UNV.SUNVNLS
//UNVCONF  DD  DISP=SHR,DSN=&CFGPRE..UNVCONF
//UNVDB    DD  DISP=SHR,DSN=UNV.UNVDB
//UNVSPOOL DD  DISP=SHR,DSN=UNV.UNVSPOOL
//UNVTRACE DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 command options
/*
```

If zFS data sets are being used instead of the default HFS data sets, the UNVDB and UNVSPOOL ddnames cannot be used to allocate the zFS data sets. The ddnames must be removed along with the DBPRE procedure parameter.

Figure 15.2 Universal Spool List for z/OS – JCL

15.2.5 Configuration Options

Table 15.3, below, identifies the Universal Spool List for z/OS configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
COMPONENT	Component identifier for which records will be selected to write.
HELP	Writes a description of the command options and their format.
LIST	Type of database from which to select record to write.
MESSAGE_LEVEL	Level of messages that will be written.
MOUNT_POINT	HFS directory in which the HFS databases allocated to ddnames UNVDB and UNVSPOOL are mounted.
VERSION	Writes the program version and copyright information.

Table 15.3 Universal Spool List for z/OS - Configuration Options

15.2.6 Command Line Syntax

Figure 15.4, below, illustrates the command line syntax — using the command line, long form of the configuration options — of Universal Spool List for z/OS.

```
uslist
[-list {ubroker|ucmd|urmtcfg|stdin|stderr|stdout}]
[-component cid]
[-mount_point directory]
[-level {audit|info|warn|error}]

uslist
{ -help | -version }
```

Figure 15.3 Universal Spool List for z/OS - Command Line Syntax

15.2.7 zFS Support

Universal Spool List (**USLIST**) and Universal Spool Remove (**USLRM**) obtain the HFS data set names from **UNVDB** and **UNVSPOOL** ddnames. zFS data sets cannot be provided via ddnames.

To obtain the zFS or HFS data set names, **USLIST** and **USLRM** allocate and parse the Universal Broker's configuration member, **UBRCFG00**, for the UNIX_DB_DATA_SET and UNIX_SPOOL_DATA_SET options. If these options are not found, **USLIST** and **USLRM** assume that HFS data sets are being used and refer the **UNVDB** and **UNVSPOOL** ddnames for the HFS data set names.

Note: Previous versions of **USLIST** and **USLRM** did not allocate the Universal Broker's configuration file.

USLIST and **USLRM** have always mounted the HFS data sets if they were not already mounted. This is the case with zFS data sets as well.

The Universal Products configuration PDSE is allocated to **UNVCONF** in the **USLLSPRC** and **USLRMPRC** JCL procedures in the **SUNVSAMP** library. The only other changes to **USLLSPRC** and **USLRMPRC** is the addition of a comment that the **UNVDB** and **UNVSPOOL** ddnames must be commented out to use zFS data sets.

USLIST and **USLRM** now write messages UNV2264I and UNV2265I, which provide information on the Broker's database and spool, respectively. The messages also provide the file system type, data set name, and mount point.

15.3 Universal Spool List for Windows and UNIX

This section describes Universal Spool List for the Windows and UNIX operating systems.

The Universal Spool List utility can be used to read the databases listed in Table 15.4.

Database Name	Database File Name
Universal Broker Component Database	bcomponent.db
Universal Server Component Database	scomponent.db
Universal Server Spool Databases	spool.stdin.*COMPID*.db spool.stdout.*COMPID*.db spool.stderr.*COMPID*.db
Universal Event Monitor Event Definition Database	ueme.db
Universal Event Monitor Event Handler Database	uemh.db
Universal Event Monitor Spool Database	uems.db

Table 15.4 Universal Spool List for Windows and UNIX - Databases

The *COMPID* in the Universal Server Spool Databases file name in this table is the component ID assigned to the Server instance.

UNIX

By default, the database files are located in the **/var/opt/universal/spool** directory.

The program file is located in the Universal Spool installation directory **bin** directory, which defaults to **/opt/universal/uspool-3.2.0/bin**.

Windows

By default, the database files are stored in the **C:\Program Files\Universal\spool** directory.

The Universal Spool List program file, **uslist.exe**, is located in the **bin** subdirectory of the Universal Spool installation directory, which defaults to **C:\Program Files\Universal\uspool**.

15.3.1 Configuration Options

Table 15.5, below, identifies the Universal Spool List for Windows and UNIX configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
BROKER_SPOOL_DIR	Directory location in which the Universal Broker Component database is located.
COMPONENT	Component identifier for which records will be selected to write.
HELP	Writes a description of the command options and their format.
ID	Lists the contents of a specific record from the Universal Event Monitor event definition, event handler, or spool databases.
LIST	Type of database from which to select record to written.
MESSAGE_LEVEL	Level of messages that will be written.
UCMD_SPOOL_DIR	Directory location in which the Universal Server Component database is located.
VERSION	Writes the program version and copyright information.

Table 15.5 Universal Spool List for Windows and UNIX - Configuration Options

15.3.2 Command Line Syntax

Figure 15.4, below, illustrates the command line syntax — using the command line, long form of the configuration options — of Universal Spool List for Windows and UNIX.

```
uslist
[-list {ubroker|ucmd|ueme|uemh|uems|urmtcfg|stdin|stderr|stdout}]
[-component cid]
[-id id]
[-brokerspooldir directory]
[-ucmdspooldir directory]
[-level {audit|info|warn|error}]

uslist
{ -help | -version }
```

Figure 15.4 Universal Spool List for Windows and UNIX - Command Line Syntax

15.4 Universal Spool List for OS/400

This section describes Universal Spool List for OS/400 operating system.

The Universal Spool List utility can be used to read the databases listed in Table 15.6, below.

Database Name	Database File Name
Universal Broker Component Database	**UBR_CMP_DB**
Universal Server Component Database	**SRV_CMP_DB**
Universal Server Spool Databases *	**SIcompid** (STDIN) **SOcompid** (STDIN) **SEcompid** (STERR)
Universal Management Console Remote Configuration Database	UNVCFG_DB
The **compid** in the Universal Server Spool Databases file names in this table is the component ID (in hexadecimal) assigned to the Server instance.	

Table 15.6 Universal Spool List for OS/400 - Databases

The spool files are located in library **UNVSPL320**.

15.4.1 Configuration Options

Table 15.7, below, identifies the Universal Spool List for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
COMPONENT	Component identifier for which records will be selected to write.
ID	Lists the contents of a specific record from the Universal Event Monitor event definition, event handler, or spool databases.
LIST	Type of database from which to select record to written.
MESSAGE_LEVEL	Level of messages that will be written.
VERSION	Writes the program version and copyright information.

Table 15.7 Universal Spool List forOS/400 – Configuration Options

15.4.2 Command Line Syntax

Figure 15.5, below, illustrates the command line syntax — using the ULSTSE parameter form of command line options — of Universal Spool List for OS/400.

```
ULSTSE
[LIST(*{ubroker|ucmd|urmtcfg|stdin|stderr|stdout})]
[COMPONENT(cid)]
[ID(id)]
[LEVEL(*{audit|info|warn|error})]

ULSTSE
VERSION(*{yes|no})
```

Figure 15.5 Universal Spool List for OS/400 – Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

15.5 Universal Spool List Output

15.5.1 Universal Broker Component

Table 15.8, below, identifies the information written for a requested Universal Broker component.

Field Name	Description
ID	Component identifier.
Name	Component definition name. Either **`ucmd`** or **`uctl`**.
Desc	Component description field from the component definition.
Version	Component version and build level.
State	Component state.
Cmd ID	Command identifier provided by the manager.
Comm State	Component communication state.
Comp State Time	Date and time the component entered the communication state.
Restartable	Specification for whether or not the component is restartable (manager fault tolerant).
Srv PID	Component's process identifier.
Srv Start Time	Components start date and time.
Srv End Time	Component's end date and time.
Srv Exit Code	Component's exit code if its status is not RUNNING.
Srv Exit Status	Component's execution status.
Mgr UID	Manager's user identifier.
Mgr Work ID	Manager's work identifier.
Mgr Host Name	Manager's TCP/IP host name on which it's executing.
Mgr Port	Manager's TCP/IP port number from which it connected.

Table 15.8 Universal Spool List Output - Universal Broker Component

15.5.2 Universal Broker Component List

Table 15.9, below, identifies the columns of data in a Universal Broker component list.

Column Name	Description
ID	Component Identifier
NAME	Component name. Either **ucmd** or **uctl**.
CST	Component's communication state. Communication state values are • COM Component is completed. • DIS Communication link between the component and the manager is disconnected. The status of the manager is unknown. The network fault tolerant protocol is being used. • EST Communication link between the component and the manager is established. This is the normal operating mode. • ORH Component is executing, but the manager has terminated. The manager is orphaned. The component was started with manager fault tolerance and is waiting for a manager restart. The user process is still executing. • PEN Component has completed its work and is waiting for a manager to restart to receive the user process spool files and exit status. The component was started with manager fault tolerance. • RCG Component is in the middle of reconnecting the manager. The network fault tolerant protocol is being used. • RSG Component is in the middle of restarting with a manager. The component was started with manager fault tolerance. • RSA Manager restart request has been accepted. The manager and the component will be reestablishing their communication links. • STR Component is starting. The component usually remains in this state for a short period of time unless they are executing with manager fault tolerance and the manager is redirecting a large stdin file.
MGR-WORK-ID	Manager's work identifier. The work ID format depends on the system type on which the manager is executing.
COMMAND-ID	Command identifier specified by the manager.

Table 15.9 Universal Spool List Output - Universal Broker Component List

15.5.3 Universal Command Server Component

Table 15.10 identifies the information in a requested Universal Command Server component.

Field Name	Description
ID	Component identifier.
Cmd Line	User command the manager requested to be executed.
User ID	User identifier with which the user command is executing.
Desc	Component description.
Comm State	Component communication state.
Comp State Time	Date and time the component entered the communication state.
Restartable	Whether the component is restartable (manager fault tolerant).
Spool Retention	Number of days to retain the spool files after the component goes into completed state.
Comp Retention	Number of days to retain the component record after the component goes into completed state.
PID	User command's process identifier.
Start Time	User process's start date and time.
End Time	User process's end date and time.
Exit Code	User process's exit code if it is not RUNNING.
Exit Status	User process's exit status.
Script File	Script file being executed by the Server.

Table 15.10 Universal Spool List - Universal Command Server Component

15.5.4 Universal Command Server Component List

Table 15.11, below, identifies the columns of data in a Universal Command Server component record.

Column Name	Description
ID	Component Identifier.
CST	Component's communication state. (See **CST** in Table 15.9 Universal Spool List Output - Universal Broker Component List.)
USER-ID	Local user account with which the user process is being executed.
COMMAND	Command which the manager requested to be executed.

Table 15.11 Universal Spool List Output - Universal Command Server Component List

15.5.5 Universal Event Monitor Event Definition

Table 15.12 identifies the information that is displayed for a requested Universal Event Monitor event definition.

Field Name	Description
Event ID	Event definition identifier.
Event Type	Type of system event that the event definition is responsible for detecting and monitoring. The following event types are supported: • FILE Instructs UEM to detect the creation of a file and track its progress until it meets its specified completion criteria.
Component Name	Name of the event-driven Universal Event Monitor Server component to which the event definition is assigned. This is the UEM Server component responsible for monitoring the event.
Enabled	Indicates whether the event definition is currently recognized by its assigned UEM Server. An enabled event definition will be monitored as long as the current date and time fall within the activation and inactivation times. A disabled event definition will never become active and will never be monitored, unless it is explicitly enabled.
Active	Indicates whether the event definition is currently being monitored by its assigned UEM Server. An event definition must be enabled before it will be made active. The event will be made inactive once its inactivation time elapses.
Activation Time	Date and time at which the assigned UEM Server component will begin monitoring this event.
Inactivation Time	Date and time at which the assigned UEM Server component will stop monitoring this event.
Tracking Interval	Frequency, in seconds, with which UEM will test for the completion of any system occurrence detected for this event.
Triggered Handler	ID of a record stored in the event handler database that should be executed whenever the processing state for an occurrence of this event is set to TRIGGERED.
Expired Handler	ID of a record stored in the event handler database that should be executed whenever the processing state for this event is set to EXPIRED.
Rejected Handler	ID of a record stored in the event handler database that should be executed whenever the processing state for an occurrence of this event is set to REJECTED.
Handler Options	Parameters that UEM adds to the command line used to execute the event handler process. The event handler process receives these parameters as command line options.
Last Modified On	Date and time the event definition record was last updated.
Last Modified By	Name of the user account that last updated the event definition record.

Table 15.12 Universal Spool List - Universal Event Monitor Event Definition

15.5.6 Event Type-Specific Fields

The following sections describe the event definition fields that vary depending on the value of the Event Type parameter.

FILE Event Definitions

Table 15.13, below, identifies the fields that are displayed for events with an event type of **FILE**.

Field Name	Description
File Specification	File whose creation should be detected and whose progress should be tracked by UEM.
Minimum File Size	Smallest size a file must be before it is considered complete by UEM.
Rename File	Indication of whether or not the file will be renamed by UEM whenever the processing state of the tracked event occurrence is set to TRIGGERED.
Rename Specification	Format that UEM should use when renaming a file whose event occurrence has been set to the TRIGGERED state.

Table 15.13 FILE Event Fields

15.5.7 Universal Event Monitor Event Definition List

Table 15.14, below, identifies the items for which values are defined in a Universal Event Monitor Event Definitions list.

Column Name	Description
EVENT ID	Event Definition Identifier.
TYPE	Type of system event that the event definition is responsible for detecting and monitoring. For a complete list of supported event types, see **Event Type** in Table 15.12 Universal Spool List - Universal Event Monitor Event Definition.
ENABLED	Indication of whether or not the event definition currently is being processed by its assigned UEM Server.
ACTIVE	Indication of whether or not the event definition currently is being monitored by its assigned UEM Server.

Table 15.14 Universal Spool List - Universal Event Monitor Event Definition List

15.5.8 Universal Event Monitor Event Handler

Table 15.15 identifies the information displayed for a requested Universal Event Monitor event handler.

Field Name	Description
Handler ID	Event Handler Identifier.
Handler Type	Process which is executed on behalf of the event handler. The following process types are supported: • CMD Indicates the record contains the name of an application, along with all of its required command line parameters, that is to be executed on behalf of the event handler. • SCRIPT Indicates the record contains a set of one or more system commands that are to be executed as a single script on behalf of the event handler.
Max Acceptable Return Code	Highest value an event handler process may return to still be considered as having executed successfully.
User ID	ID of the user account in whose security context the event handler process will be executed.
Command	If the value of the Handler Type parameter is **CMD**, this field shows the command to execute. This field will not be shown if the value of the Handler Type parameter is **SCRIPT**.
Script Statements	If the value of the Handler Type parameter is **SCRIPT**, this field marks the beginning of the system commands that will be executed as a script. This field will not be shown if the value of the Handler Type parameter is **CMD**.
Script Type	Type of script statements to execute when the value of the Handler Type parameter is **SCRIPT**. This field will not be shown if the value of the Handler Type parameter is **CMD**.
Last Modified On	Date and time the event handler record was last updated.
Last Modified By	Name of the user account that last updated the event handler record.

Table 15.15 Universal Spool List - Universal Event Monitor / Event Handler

15.5.9 Universal Event Monitor Event Handler List

Table 15.16, below, identifies the items for which values are defined in a Universal Event Monitor Event Handlers list.

Column Name	Description
HANDLER ID	Event Handler Identifier.
TYPE	Describes the process which is executed on behalf of the event handler. For a complete list of supported process types, see **`Handler Type`** in Table 15.15 Universal Spool List - Universal Event Monitor / Event Handler.

Table 15.16 Universal Spool List - Universal Event Monitor Event Handler List

15.5.10 Universal Event Monitor Spool List

Table 15.17 identifies the items for which values are listed in Universal Event Monitor Spool List.

Column Name	Description
SERIAL NO	A sequential number that is assigned to each record as it is added to the database. This number serves to uniquely identify each occurrence of a given event definition.
EVENT ID	The ID of the event definition responsible for the detection and monitoring of the event occurrence recorded by this spool record.
PRC STATE	The processing state of the event occurrence. For a complete list of possible values, see the description of the Processing State parameter, below.
HANDLER ID	The ID of an event handler executed whenever the processing state of an event or an event occurrence enters the TRIGGERED, REJECTED, or EXPIRED state.
EXIT CODE	The value returned by the process executed on behalf of an event handler.
EXIT STATUS	Indicates whether the event handler process ended normally or whether it was terminated unexpectedly.
HANDLER STATUS	Indicates the outcome of event handler processing. For a list of possible values, see **`Handler Status`** in Table 15.18 Universal Spool List - Universal Event Monitor Spool Record.

Table 15.17 Universal Spool List - Universal Event Monitor Spool List

15.5.11 Universal Event Monitor Spool Record

Table 15.18 identifies the information displayed for a requested Universal Event Monitor spool record.

Field Name	Description
Serial No	Sequential number that is assigned to each record as it is added to the database. This number serves to uniquely identify each occurrence of a given event definition.
Event ID	ID of the event definition responsible for the detection and monitoring of the event occurrence recorded by this spool record.
Component Name	Name of the event-driven Universal Event Monitor Server component to which the event definition is assigned. This is the UEM Server component responsible for monitoring and processing the event.
Component Description	Description of the UEM Server component identified by Component Name.
Component Version	Version of the UEM Server component identified by Component Name.
Component ID	Value that uniquely identifies the instance of the UEM Server component that processed the event occurrence.
Event Type	Type of system event that the event definition is responsible for detecting and monitoring.
System Object	System event detected and monitored by the event occurrence.
Processing State	Processing state of the event occurrence. The following values are used: • TRACKING Indicates that an occurrence of a system event described by an event definition was detected, but has not yet met the completion criteria set forth by the event definition and by UEM's application logic. • TRIGGERED Indicates that an occurrence of a system event described by an event definition was detected and has completed. If a triggered event handler was specified in the event definition, that handler's process will be executed and the event handler's ID will be shown in the Handler ID field. • REJECTED Indicates than an occurrence of a system event described by an event definition was detected, but failed to complete before the date and time specified in the event definition's Inactivation Time parameter. If a rejected event handler was specified in the event definition, that handler's process will be executed and the event handler's ID will be shown in the Handler ID field. • EXPIRED Indicates that no occurrence of the system event described by an event definition was detected before the event's Inactivation Time elapsed. If an expired event handler was specified in the event definition, that handler's process will be executed and the event handler's ID will be shown in the Handler ID field. • ERROR Indicates an error occurred while processing the event occurrence.
Handler ID	ID of an event handler executed whenever the processing state of an event or an event occurrence enters the TRIGGERED, REJECTED, or EXPIRED state.

Field Name	Description
User Command	Command executed on behalf of the event handler when the Handler Type is CMD. If the Handler Type is SCRIPT, this field contains no value.
Process ID	ID of the process executed by UEM on behalf of the event handler.
User ID	Name of the user account in whose security context the event handler process was executed.
Start Time	Date and time the event handler process started.
End Time	Date and time the event handler process ended.
Exit Code	Value returned by the event handler process.
Exit Status	Indicates whether the event handler process ended normally or whether it terminated unexpectedly.
Handler Status	Indicates the outcome of event handler processing. The following values are used: • FAILED The event handler process finished abnormally, or ended normally with an exit code greater than the maximum acceptable return code specified in the event handler record. • NO HANDLER No event handler was specified for the event's processing state. • NOT AUTHORIZED An attempt to execute the event handler process failed because the user ID or password specified for the event handler was incorrect. • NOT FOUND The ID of an event handler record specified for a particular processing state was not found in the event handler database. • SHUTDOWN The Universal Event Monitor Server was stopped while the event handler process was running. • SUCCESSFUL The event handler process completed normally and exited with a value that was less than or equal to the maximum acceptable return code specified in the event handler record. • UNRECOVERABLE Information for the event handler process could not be recovered.
Last Modified On	Date and time the spool record was last updated.

Table 15.18 Universal Spool List - Universal Event Monitor Spool Record

15.6 Examples of Universal Spool List

This section contains examples demonstrating the use of Universal Spool List.

The following list provides a link to each example.

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

15.6.1 List Universal Broker Database

Figure 15.6, below, illustrates how to execute USLIST with all defaults.

Note: No parameters are required to issue USLIST.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslist
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 15.6 Universal Spool List - List Universal Broker Database

Since no parameters are supplied, by default, this example 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. If executing on UNIX, change the directory path to the **`uslist`** executable.

Command Line Options

No command line options are used in this example.

15.6.2 List Universal Server Database Records

Figure 15.7, below, illustrates how to specify options for the USLIST command.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslist -list UCMD –ucmdspooldir “c:\program files\universal\spool2”

/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 15.7 Universal Spool List - List Universal Server Database Records

This example lists the contents of the Universal Command Server Component database. A summary of all records is written.

This example specifies the directory location in which the Universal Server Component database is located. If the directory contains spaces, it must be enclosed in double (**"**) quotes. If executing on UNIX, change the directory path to the **uslist** executable.

Command Line Options

The command line options used in this example are:

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

15.6.3 List Broker Detail for a Component

Figure 15.8, below, illustrates how to list the broker detail for a specific component ID.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslist -component 123456789
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 15.8 Universal Spool List - List Broker Detail for a Component

Since the list parameter is not supplied, by default, this example lists 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**. If executing on UNIX, change the directory path to the **uslist** executable.

Command Line Options

The command line option used in this example is:

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

15.6.4 List Standard Out for a Component

Figure 15.9, below, illustrates how to list the standard output spool file for a specific component ID.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslist -list STDOUT -component 123456789
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 15.9 Universal Spool List - List Standard Out for a Component

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

Command Line Options

The command line options used in this example are:

Option	Description
-list	Specifies the type of database from which to select records to write.
-component	Specifies the component identifier for which records will be selected to write.

Chapter 16
Universal Spool Remove

16.1 Overview

Universal Spool Remove (**USLRM**) utility provides the ability to remove component records from the Universal Command and Universal Event Monitor (UNIX and Windows only) Spool databases. Universal Spool Remove must be executed on the system upon which the database is located.

By default, spool records are not retained after they no longer are needed. Accordingly, it is not anticipated that the spool databases will become too large. However, on occasion, some records may not be cleaned up, making it necessary to remove them with the Universal Spool Remove utility.

16.1.1 Prerequisite to Running Universal Spool Remove

Before attempting to remove any records using Universal Spool Remove, ensure that the Universal Broker is not running on the local system.

While it is active, the Universal Broker, in its role as a local database administrator, actually “owns” and maintains an open reference to the spool databases. Any changes made to these databases outside of the Broker are not committed to the database while this reference is open. If Universal Spool Remove removes a spool record while the Broker is running, that same record will “reappear” the next time that the Broker commits an update to the database (for example, a new server component is started and recorded in the spool).

All spool records that are deleted as part of the regular component clean-up are permanent because those deletions are done via the Broker.

The functions that Universal Spool List program provide are required for possible database clean-up (see Chapter 16 Universal Spool List).

16.2 Usage

The Universal Spool Remove utility removes all records for a specified component ID from the Universal Spool databases.

Any errors encountered while records are being removed from a database will be reported, but will not result in the program being stopped.

Universal Spool Remove removes records as specified by the command options.

16.2.1 Security

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

16.3 Universal Spool Remove for z/OS

This section describes Universal Spool Remove for the z/OS operating system.

16.3.1 Databases

Universal Spool databases are implemented as HFS data sets. The HFS data sets UNVDB and UNVSPOOL contain an HFS file system that contains the Universal Spool database files.

Table 16.1, below, identifies the database files and the HFS data sets in which they reside.

Database Name	Data Set	File Name
Universal Broker Component Database	UNVDB	bcomponent.db
Universal Server Component Database	UNVDB	scomponent.db
Universal Server Spool Databases	UNVSPOOL	spool.stdin.*COMPID*.db spool.stdout.*COMPID*.db spool.stderr.*COMPID*.db
Note: In the Universal Server Spool Databases file names, *COMPID* is the component ID assigned to the Server instance.		

Table 16.1 Universal Spool Remove for z/OS - Universal Spool Databases

16.3.2 JCL Procedure

Figure 16.1, below, illustrates the Universal Spool Remove for z/OS JCL procedure (**USLRMPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//USLRMPRC PROC UPARM=,               -- USLRM options
//             UNVPRE=#SHLQ.UNV,
//             CFGPRE=#PHLQ.UNV,
//             DBPRE=#PHLQ.UNV
//*
//PS1      EXEC PGM=USLRM,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UNVPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UNVPRE..SUNVNLS
//UNVCONF  DD  DISP=SHR,DSN=&CFGPRE..UNVCONF
//UNVDB    DD  DISP=SHR,DSN=&DBPRE..UNVDB
//UNVSPOOL DD  DISP=SHR,DSN=&DBPRE..UNVSPOOL
//UNVTRACE DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

If zFS data sets are being used instead of the default HFS data sets, the UNVDB and UNVSPOOL ddnames cannot be used to allocate the zFS data sets. The ddnames must be removed along with the DBPRE procedure parameter.

Figure 16.1 Universal Spool Remove for z/OS – JCL Procedure

16.3.3 DD Statements used in JCL Procedure

Table 16.2, below, describes the DD statements used in the Universal Spool Remove for z/OS JCL procedure illustrated in Figure 16.2.

ddname	Description
STEPLIB	Load library in which program USLRM program is located.
UNVNLS	Universal National Language Support library.
UNVCONF	Universal Products configuration library.
UNVDB	Universal Broker Database HFS data set.
UNVSPOOL	Universal Spool Database HFS data set.
UNVTRACE	Application trace ddname.
SYSOUT	USLRM standard error ddname.
SYSPRINT	USLRM standard output ddname.

Table 16.2 Universal Spool Remove for z/OS – DD Statements in JCL Procedure

16.3.4 JCL

Figure 16.2, below, illustrates the Universal Spool Remove for z/OS JCL.

```
//STEP1    EXEC PGM=USLRM
//STEPLIB  DD  DISP=SHR,DSN=UNV.SUNVLOAD
//UNVNLS   DD  DISP=SHR,DSN=UNV.SUNVNLS
//UNVCONF  DD  DISP=SHR,DSN=&CFGPRE..UNVCONF
//UNVDB    DD  DISP=SHR,DSN=UNV.UNVDB
//UNVSPOOL DD  DISP=SHR,DSN=UNV.UNVSPOOL
//UNVTRACE DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 command options
/*
```

If zFS data sets are being used instead of the default HFS data sets, the UNVDB and UNVSPOOL ddnames cannot be used to allocate the zFS data sets. The ddnames must be removed along with the DBPRE procedure parameter.

Figure 16.2 Universal Spool Remove for z/OS – JCL

16.3.5 Configuration Options

Table 16.3, below, identifies the Universal Spool Remove for z/OS configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
COMPONENT	Component identifier for which records will be removed.
HELP	Writes a description of the command options and their format.
MESSAGE_LEVEL	Level of messages that will be written.
MOUNT_POINT	HFS directory in which the HFS databases allocated to ddnames UNVDB and UNVSPOOL are mounted.
VERSION	Writes the program version and copyright information.

Table 16.3 Universal Spool Remove for z/OS - Configuration Options

16.3.6 Command Line Syntax

Figure 16.4, below, illustrates the command syntax — using the command line, long form of the configuration options — of Universal Spool Remove for z/OS.

```
uslrm
-component cid
[-mount_point dir]
[-level {audit|info|warn|error}]

uslrm
{ -help | -version }
```

Figure 16.3 Universal Spool Remove for z/OS - Command Line Syntax

16.3.7 zFS Support

Universal Spool Remove (**USLRM**) and Universal Spool List (**USLIST**) obtain the HFS data set names from **UNVDB** and **UNVSPOOL** ddnames. zFS data sets cannot be provided via ddnames.

To obtain the zFS or HFS data set names, **USLRM** and **USLIST** allocate and parse the Universal Broker's configuration member, **UBRCFG00**, for the UNIX_DB_DATA_SET and UNIX_SPOOL_DATA_SET options. If these options are not found, **USLRM** and **USLIST** assume that HFS data sets are being used and refer the **UNVDB** and **UNVSPOOL** ddnames for the HFS data set names.

Note: Previous versions of **USLRM** and **USLIST** did not allocate the Universal Broker's configuration file.

USLRM and **USLIST** have always mounted the HFS data sets if they were not already mounted. This is the case with zFS data sets as well.

The Universal Products configuration PDSE is allocated to **UNVCONF** in the **USLLSPRC** and **USLRMPRC** JCL procedures in the **SUNVSAMP** library. The only other changes to **USLLSPRC** and **USLRMPRC** is the addition of a comment that the **UNVDB** and **UNVSPOOL** ddnames must be commented out to use zFS data sets.

USLRM and **USLIST** now write messages UNV2264I and UNV2265I, which provide information on the Broker's database and spool, respectively. The messages also provide the file system type, data set name, and mount point.

16.4 Universal Spool Remove for Windows and UNIX

Table 16.4, below, identifies the databases from which Universal Spool Remove can remove records.

Database Name	Default File Name
Universal Broker Component Database	bcomponent.db
Universal Server Component Database	scomponent.db
Universal Server Spool Databases	spool.stdin.*COMPID*.db spool.stdout.*COMPID*.db spool.stderr.*COMPID*.db
Universal Event Monitor Spool Database	uems.db
Note: In the Universal Server Spool Databases file names, *COMPID* is the component ID assigned to the Server instance.	

Table 16.4 Universal Spool Remove for Windows and UNIX - Universal Spool Databases

UNIX

By default, the database files are located in the **/var/opt/universal/spool** directory.

The program file is located in the Universal Spool installation directory **bin** directory, which defaults to **/opt/universal/uspool/bin**.

Windows

By default, the database files are stored in the **C:\Program Files\Universal\spool** directory.

The Universal Spool Remove program file, **uslrm.exe**, is located in the **bin** subdirectory of the Universal Spool installation directory, which defaults to **C:\Program Files\Universal\uspool**.

16.4.1 Configuration Options

Table 16.5, below, identifies the Universal Spool Remove for Windows and UNIX configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
BROKER_SPOOL_DIR	Directory location in which the Universal Broker Component database is located.
COMPONENT	Component identifier for which records will be removed.
HELP	Writes a description of the command options and their format.
MESSAGE_LEVEL	Level of messages that will be written.
UCMD_SPOOL_DIR	Directory location in which the Universal Server Component database is located.
UEM_SERIALNO	Serial number of the Universal Event Monitor spool database record to remove.
VERSION	Writes the program version and copyright information.

Table 16.5 Universal Spool Remove for Windows and UNIX - Configuration Options

16.4.2 Command Line Syntax

Figure 16.4, below, illustrates the command line syntax — using the command line, long form of the configuration options — of Universal Spool Remove for Windows and UNIX.

```
uslrm
{ -component cid | -uem_serialno serno }
[-brokerspooldir dir]
[-ucmdspooldir dir]
[-level {audit|info|warn|error}]

uslrm
{ -help | -version }
```

Figure 16.4 Universal Spool Remove for Windows and UNIX - Command Line Syntax

16.5 Universal Spool Remove for OS/400

This section describes Universal Spool Remove for OS/400 operating system.

The Universal Spool Remove utility can be used to read the databases listed in Table 16.6, below.

Database Name	Database File Name
Universal Broker Component Database	**UBR_CMP_DB**
Universal Server Component Database	**SRV_CMP_DB**

Table 16.6 Universal Spool Remove for OS/400 – Databases

The spool files are located in library **UNVUSL320**.

16.5.1 Configuration Options

Table 16.7, below, identifies the Universal Spool Remove for OS/400 configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
COMPONENT	Component identifier for which records will be removed.
MESSAGE_LEVEL	Level of messages that will be written.
VERSION	Writes the program version and copyright information.

Table 16.7 Universal Spool Remove forOS/400 – Configuration Options

16.5.2 Command Line Syntax

Figure 16.5, below, illustrates the command line syntax — using the URMVSE parameter form of command line options — of Universal Spool Remove for OS/400.

```
URMVSE
[COMPONENT(cid)]
[LEVEL(*{audit|info|warn|error})]

URMVSE
VERSION(*{yes|no})
```

Figure 16.5 Universal Spool Remove for OS/400 – Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

16.6 Example of Universal Spool Remove

This section contains examples demonstrating the use of Universal Spool Remove.

The following list provides a link to each example.

- Remove Component Records
- Change Broker Database Directory

```
uslrm
-component cid
[-mount_point dir]
[-level {audit|info|warn|error}]

uslrm
{ -help | -version }
```

16.6.1 Remove Component Records

Figure 16.6, below, illustrates how to execute Universal Spool remove with defaults.

The only required parameter is the component ID. The USLIST utility can be executed to find specific component IDs.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslrm -component 123456789
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 16.6 Universal Spool Remove - Remove Component Records

All Universal Products database records will be removed for component **123456789**. If executing on UNIX, change the directory path to the USLRM executable.

Command Line Options

The command line options used in this example are:

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

16.6.2 Change Broker Database Directory

Figure 16.7, below, illustrates how to specify a database directory other than the default for the USRLM command.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD *
cd c:\program files\universal\uspool\bin
uslrm -c 123456789 -brokerspooldir "c:\program files\universal\spool2"
/*
//SYSIN    DD  *
 -host dallas
 -script SCRIPTDD
 -u joe
 -pwd abcdefg
/*
```

Figure 16.7 Universal Spool Remove - Remove Component Records

All Universal Products database records will be removed. This example 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.

Command Line Options

The command line options used in this example are:

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

Chapter 17
Universal Submit Job

17.1 Overview

The Universal Submit Job (USBMJOB) utility is a command for the OS/400 environment that encapsulates the IBM Submit Job (SBMJOB) command.

USBMJOB builds on the functionality of SBMJOB by providing a job submission command that better suits the needs of a remote user issuing OS/400 commands via Universal Command.

Note: Users never should call USBMJOB directly.

17.1.1 Functions

USBMJOB performs four main functions:

1. Submits commands in their own easily customized job environment.
2. Monitors submitted jobs to completion and sets a return code based on the end code and message severity codes of the submitted job.
3. Provides a facility for the remote handling of inquiry messages generated by the submitted job. Inquiry messages are received and replied to from the z/OS console.
4. Returns spooled output generated by the submitted job on standard output. The joblog generated by the submitted job is written to standard error.

17.2 Usage

Universal Submit Job (USBMJOB) submits a user command in its own job. USBMJOB supports all of the command parameters offered by the SBMJOB command that make sense for an unscheduled job submitted from a batch environment. This enables the user to fully customize the job environment for the user command. Internally, the SBMJOB command is called to submit the user job.

USBMJOB remains active for the duration of the submitted job. USBMJOB continuously monitors the state of the submitted job at a user-defined polling interval. In addition to monitoring for job completion, USBMJOB can detect when the submitted job is waiting for a reply to an inquiry message.

USBMJOB provides the option for inquiry messages generated from the submitted job to be sent to a remote z/OS console. Replies received from the z/OS console are sent as reply messages to the corresponding inquiry message.

When the submitted job completes, USBMJOB writes the joblog for the submitted job to standard error. The spooled output generated by the submitted job is written to standard output.

After USBMJOB has finished processing the submitted job, it completes by issuing an escape message to the external message queue. The escape message sets the return code for the USBMJOB command. The severity code of the escape message indicates the return code.

If the user job submitted by USBMJOB completes normally (end code < 20), the severity code for the USBMJOB escape message will be 0. If the user job submitted by USBMJOB completes abnormally (end code > 10), the severity code for the USBMJOB escape message will be set to the highest severity code generated by the submitted job.

By issuing an escape message with a severity code correlated with the submitted job's end code/highest severity code, USBMJOB allows the Universal Command Server to pick up the severity code. This, in turn, allows the USBMJOB command to propagate its return code to the Universal Command Server.

Two helper commands - supplied by Stonebranch, Inc. - are called internally by USBMJOB:

- Universal Job initializer (UJOBINIT)
- Universal Message Handler (UMSGHNDLR)

Universal Job initializer (UJOBINIT) is called from within the job submitted by USBMJOB. UJOBINIT performs initialization that allows USBMJOB to redirect the joblog of the submitted job and then issues the user command.

Output

In addition to joblog redirection, USBMJOB returns the jobs spooled output to the Universal Command Manager via standard output.

When the submitted job user name and the user profile name passed to the Universal Command Manager differ, USBMJOB requires *SPLCTL authority to retrieve the spooled output. USBMJOB receives this *SPLCTL authority from the **UNVUBR320** user profile.

If *SPLCTL special authority is removed from the **UNVUBR320** user profile, USBMJOB will fail if:

1. Security is set to DEFAULT via the UNVCONF(UCMDS) configuration file and the submitted job user name and the user profile name passed to the Universal Command Manager differ.

   Example:

   ```
   ucmd -c "usbmjob cmd(dsplib abc) user(abc)" -i rmtsys -u myuser
   -w mypwd
   ```

2. Security is set to NONE via the UNVCONF(UCMDS) configuration file and a user name is specified for the submitted job.

   Example:

   ```
   ucmd -c "usbmjob cmd(dsplib abc) user(abc)" -i rmtsys
   ```

17.2.1 Configuration Options

The Universal Submit Job command performs operations specified by configuration options. The options have associated values that describe the actions to take.

The USBMJOB configuration options are separated into two categories:

1. USBMJOB-specific
2. SBMJOB encapsulated

USBMJOB-Specific Configuration Options

Table 17.1, below, identifies the USBMJOB-specific configuration options, which control the way that the submitted job is monitored and administered. Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
COMMAND	Command that runs in the submitted batch job.
COPY_SPOOL_FILES	Specification for whether or not spooled output files generated by the submitted job be copied to standard output.
ENCRYPTED_COMMAND_FILE	Name of an encrypted command file.
JOB_LOG_LIBRARY	Library into which the job log will be placed.
JOB_STATUS_POLLING_INTERVAL	Number of seconds that USBMJOB will sleep between calls to check the status of the submitted job.
KEY	Encryption key used to encrypt the encrypted command file.
REMOTE_MESSAGE_PREFIX	Text string that prefixes any remote message sent by USBMJOB.
REMOTE_REFRESH_INTERVAL	Time that a remote reply message will remain on a remote console without being replied to before it times out.
REMOTE_REPLY_COMMAND_PATH	Path (including the executable name) to the remote message handler (**uwto**).
REMOTE_REPLY_HOST	Host name of the remote system on which the **uwto** command is executed.
REMOTE_REPLY_PORT	Port of the Universal Broker on the remote system on which the Universal WTO command is executed.
REMOTE_REPLY_USER_ID	User ID for the remote system where the **uwto** command resides.
REMOTE_REPLY_USER_PWD	Password for user on the remote system where **uwto** resides.
SPECIFY_PRINT_CONTROL_CHARS	Print control characters (if any) that are to replace the spooled file's internal print control characters.
TRACE	Specification for whether or not trace information will be printed to standard error.
USE_REMOTE_REPLY_FACILITY	Specification for whether or not USBMJOB will use the remote reply facility.

Table 17.1 Universal Submit Job - USBMJOB-Specific Configuration Options

SBMJOB Encapsulated Configuration Options

The SBMJOB encapsulated configuration options have a one-to-one relationship with the IBM SBMJOB command parameters of the same name.

Option Name	USBMJOB Parameter
ALLOW DISPLAY BY WRKSBMJOB	DSPSBMJOB({***yes** | ***no**})
CODED CHARACTER SET ID	CCSID({ ***current** | ***sysval** | ***usrprf** | ***hex** | *coded_character_set_identifier* })
COPY ENVIRONMENT VARIABLES	CPYENVVAR({ ***no** | ***yes** })
COUNTRY ID	CNTRYID({ ***current** | ***sysval** | ***usrprf** | *country_id*})
CURRENT LIBRARY	CURLIB({ ***current** | ***usrprf** | ***crtdft** | *current_library_name* })
HOLD ON JOB QUEUE	HOLD({ ***jobd** | ***no** | ***yes** })
INITIAL LIBRARY LIST	INLLIBL({ ***current** | ***jobd** | ***sysval** | ***none** | *library_name...*})
INLASPGRP	INLASPGRP(*current | *jobd | *none)
INQUIRY MESSAGE REPLY	INQMSGRPY({ ***jobd** | ***rqd** | ***dft** | ***sysrpyl** })
JOB DATE	DATE({ ***jobd** | ***sysval** | *job_date* })
JOB DESCRIPTION	JOBD({ ***usrprf** | [*library/*] *job_description* })
JOB MESSAGE QUEUE FULL ACTION	JOBMSGQFL({ ***jobd** | ***sysval** | ***nowrap** | ***wrap** | ***prtwrap** })
JOB MESSAGE QUEUE MAXIMUM SIZE	JOBMSGQMX({ ***jobd** | ***sysval** | *maximum_size_of_job_message_queue* })
JOB NAME	JOB({ ***jobd** | *job_name* })
JOB PRIORITY	JOBPTY(*priority*)
JOB QUEUE	JOBQ({ ***jobd** | [*library/*] *job_queue* })
JOB SWITCHES	SWS({ ***jobd** | *switch_settings* })
LANGUAGE ID	LANGID({***current** | ***sysval** | ***usrprf** | *language_id* })
LOG CL PROGRAM COMMANDS	LOGCLPGM({***jobd** | ***no** | ***yes** })
OUTPUT PRIORITY	OUTPTY(*priority*)
OUTPUT QUEUE	OUTQ({ ***current** | ***usrprf** | ***dev** | ***jobd** | [*library/*] *output_queue* })
PRINT DEVICE	PRTDEV({ ***current** | ***usrprf** | ***sysval** | ***jobd** | *printer_device_name* })
PRINT TEXT	PRTTXT(*text*)
SORT SEQUENCE	SRTSEQ({ ***current *sysval *usrprf *hex *langidunq *langidshr** | [{ ***libl** | ***curlib** | *library_name* } /] *table_name* })
SUBMITTED FOR	SBMFOR(*job_number* / *user* / *job_name*)
SYSTEM LIBRARY LIST	SYSLIBL({ ***current** | ***sysval** })
USER	USER({ ***current** | ***jobd** | *user_name* })

Table 17.2 Universal Submit Job - SBMJOB Encapsulated Configuration Options

Note: The values for these options are passed directly to the SBMJOB command internally. Therefore, the effect these options have on a submitted job will be equal to that documented by IBM for the Submit Job command.

Refer to the documentation provided by IBM for the SBMJOB implementation being used for an accurate description of the effect that these options will have on that implementation.

17.2.2 Command Line Syntax

Figure 17.1, below, illustrates the command line syntax of Universal Submit Job.

```
USBMJOB
[CMD(command)]
[JOBLOGLIB(library)]
[POLL(seconds)]
[RMTRPY( {*yes|*no} )]
[RMTREFRESH(seconds)]
[RMTMSGPRX(prefix)]
[RMTUSER(userid)]
[RMTPWD(password)]
[ECMFILE(cmd_file) [ECMMBR(member)] [KEY(key)] ]
[RMTHOST(host)]
[RMTPORT (port)]
[MSGCMDPATH(path)]
[JOB( { *jobd | job_name })]
[JOBD( { *usrprf | [library/ ] job_description } )]
[JOBMSGQFL( {*jobd|*sysval|*nowrap|*wrap|*prtwrap} )]
[JOBQ( { *jobd | [ library/ ] job_queue } )]
[JOBPTY(priority)]
[OUTPTY(priority)]
[PRTDEV( { *current | *usrprf | *sysval | *jobd | printer_device_name } )]
[OUTQ( { *current | *usrprf | *dev | *jobd | [ library/ ] output_queue } )]
[CPYSPLF( {*yes|*no} )]
[SPLFCTLCHR ( {*none|*fcfc|*prtctl|*s36fmt} )]
[TRACE( {*yes|*no} )]
[USER( { *current | *jobd | user_name } )]
[PRTTXT(text)]
[SYSLIBL( {*current|*sysval} )]
[CURLIB( { *current | *usrprf | *crtdft | current_library_name } )]
[INLLIBL({ *current | *jobd | *sysval | *none |library_name...} )]
[LOGCLPGM( {*jobd|*no|*yes} )]
[INQMSGRPY( {*jobd|*rqd|*dft|*sysrpyl} )]
[INLASPGRP({ *current | *jobd | *none} )]
[HOLD( {*jobd|*no|*yes} )]
[DATE( {*jobd|*sysval|job_date} )]
[SWS( {*jobd|switch_settings} )]
[DSPSBMJOB( {*yes|*no} )]
[SRTSEQ( {*current*sysval*usrprf*hex*langidunq*langidshr |
         [*libl | *curlib | library/] table_name} )]
[LANGID( {*current|*usrprf|*sysval|language_id} )]
[CNTRYID( {*current|*usrprf|*sysval|country_id} )]
[CCSID( {*current|*usrprf|*sysval|*hex|coded_character_set_identifier} )]
[SBMFOR(job_number/user/job_name)]
[JOBMSGQMX( {*jobd|*sysval|*maximum_size_of_job_message_queue} )]
[CPYENVVAR( {*yes|*no} )]
```

Figure 17.1 Universal Submit Job - Command Line Syntax

Command Line Syntax Rules

Values for configuration options that contain special characters require:

- Double (**"**) quotation marks when executed from an MVS Universal Command Manager.
- Single (**'**) quotation marks when executed from an OS/400 Universal Command Manager.

For example, the following is correct when executed from a z/OS Universal Command Manager:

```
MSGCMDPATH("/usr/local/universal/bin/uwto")
```

However, the following is incorrect when executed from a z/OS Universal Command Manager; it will create a syntax error:

```
RMSGCMDPATH('/usr/local/universal/bin/uwto')
```

17.3 Remote Reply Facility

Universal Submit Job can detect when messages sent by the submitted job require a reply. If the Remote Reply Facility is turned on (RMTRPY(***yes**)), USBMJOB will send all messages requiring a reply to a remote z/OS console. Replies to the inquiry messages are received from the z/OS console and sent to the OS/400 message queue waiting for the reply.

The Remote Reply Facility requires an installation of Universal Command on the OS/400 system where the USBMJOB command will run and an installation of Universal Command on the remote z/OS system where the inquiry messages will be sent to for reply.

- Universal Command on the OS/400 must be at Universal Command 1.2.1 level 7 or greater.
- Universal Command on the remote z/OS system, where the inquiry messages will be sent, must be at level 12 or greater.

The Remote Reply Facility used by USBMJOB is comprised of a group of Stonebranch Inc. utilities that work together (Table 17.3).

Utility	Platform
Universal Submit Job	OS/400
Universal Message Handler	OS/400
Universal Command	OS/400 and z/OS
Universal UWTO	z/OS Unix System Services (USS)

Table 17.3 Remote Reply Facility Utilities

Universal Submit Job and Universal Message Handler are part of the Universal Command for OS/400 licensed product. These utilities were added to the product in Universal Command 1.2.1 level 7).

Universal WTO (UWTO) is a command line utility for the z/OS Unix System Services (USS) environment, as of level 12. The path to the UWTO command is specified on the **MSGCMDPATH** parameter of the USBMJOB command.

See Chapter 18 Universal Write-to-Operator for more information on the UWTO command.

17.4 Return Codes

The Universal Submit Job command completes by sending an escape message to the external message queue. The severity code of this escape message is the USBMJOB return code. The USBMJOB return code is determined in the following way:

When the submitted job completes, USBMJOB scans the job log, examining the severity code of all *ESCAPE, *NOTIFY, *STATUS, and function check messages.

- If the submitted job completed abnormally (end code > 10), the USBMJOB return code will be set to the highest severity code examined in the submitted job's job log.
- If the submitted job completes normally (end code < 20), the examined severity codes will be ignored and the USBMJOB return code will be set to 0.

Setting the return code with an escape message allows the Universal Command Server to pick up the return code for use in its return code processing.

The range of possible severity code values is 0 through 99.

If USBMJOB encounters a processing error, a severity code of 99 will be used, regardless of severity codes that may have been examined from the submitted job.

17.5 Example of Universal Submit Job

This section contains examples demonstrating the use of Universal Copy.

The following list provides a link to each example.

- Universal Submit Job from z/OS to OS/400
- Universal Submit Job from z/OS to OS/400 with WTOR Support
- Universal Submit Job from Windows / UNIX to OS/400

17.5.1 Universal Submit Job from z/OS to OS/400

Figure 17.1, below, illustrates the issuing of a command to the remote OS/400 as a parameter of the USBMJOB.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD SYSOUT=*
//UNVERR   DD SYSOUT=*
//SCRIPT   DD *
 ADDLIBLE lib(UNVPRD320)
 UNVPRD320/USBMJOB CMD(dsplib tuser1)
//SYSIN    DD *
 -s SCRIPT
 -host as400 -u tuser1 -w tuser1
/*
```

Figure 17.1 Universal Submit Job - z/OS to OS/400

This Universal Command manager executes the script to a host called **as400**. UserID of **tuser1** and password of **tuser1** are used for authentication. The script runs with the authority of UserID **tuser1**.

The first line of the script will add the library **UNVPRD320** to the library concatenation of user **tuser1**. The second line will execute the command **dsplib tuser1** with the USBMJOB utility.

All output created by the command will be spooled to stdout of the manager job.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-script	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
-host	Directs the command to a computer with a host name of **as400**.
-u	Specifies the remote user ID with which to execute the command.
-w	Specifies the password for the remote user ID.

USBMJOB Options

The USBMJOB option used in this example is:

Option	Description
CMD	Specifies a command that runs in the submitted batch job. The command can be a maximum of 3000 characters.

17.5.2 Universal Submit Job from z/OS to OS/400 with WTOR Support

Figure 17.2, below, illustrates the issuing of a command to the remote OS/400 as a parameter of the USBMJOB.

The native SBMJOB parameters can be specified as part of the USBMJOB command. This process is set up so that the Universal Submit Job detects when a message sent by the submitted job requires a reply. The USBMJOB then executes the Universal WTO utility on USS to obtain a reply from the z/OS system.

```
//stepname EXEC UCMDPRC
//UNVOUT   DD SYSOUT=*
//UNVERR   DD SYSOUT=*
//SCRIPT   DD *
 ADDLIBLE lib(UNVPRD320)
 UNVPRD320/USBMJOB CMD(dsplib tuser1) +
    RMTRPY(*YES) +
    RMTREFRESH(60) +
    RMTMSGPRFX('TESTPRFX') +
    RMTHOST(OS390NEW) +
    MSGCMDPATH("/usr/local/universal/bin/uwto") +
    RMTUSER(userid)        +
    RMTPWD(pwd)
//SYSIN   DD *
 -s SCRIPT
 -host as400 -u tuser1 -w tuser1
/*
```

Figure 17.2 Universal Submit Job - z/OS to OS/400 with WTOR Support

This Universal Command Manager executes the script to a host called **AS400**. UserID of **tuser1** and password of **tuser1** are used for authentication. The script runs with the authority of UserID **tuser1**.

The first line of the script will add the library **UNVPRD320** to the library concatenation of user **tuser1**. The second line will execute the command **dsplib tuser1** with the USBMJOB utility. All output created by the command will be spooled to stdout of the manager job. The Remote Reply Facility is turned on; therefore, USBMJOB will send all messages requiring a reply to the remote z/OS console for **OS390NEW**. Replies to the inquiry messages are received from the z/OS console and sent to the AS/400 message queue waiting for the reply via the USS utility UWTO. UWTO is executed with the authority of **userid** and **pwd**. The WTO is prefixed with **TESTPRFX**. If a response is not received within 60 seconds, the WTOR will be deleted and a new one sent. The UWTO executable is found on the USS system in **/usr/local/universal/bin/uwto**.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
`-script`	Specifies the DD from which to read a script file. The script file is sent to the remote system for execution.
`-host`	Directs the command to a computer with a host name of **`as400`**.
`-u`	Specifies the remote user ID with which to execute the command.
`-w`	Specifies the password for the remote user ID.

USBMJOB Options

The USBMJOB options used in this example are:

Option	Description
`CMD`	Specifies a command that runs in the submitted batch job. The command can be a maximum of 3000 characters.
`RMTRPY`	Specifies a time interval (in seconds). The time interval will control how long a remote reply message will remain on a remote console without being replied to before it will time out.
`RMTREFRESH`	The remote message prefix allows the user to specify a text string up to 12 characters in length that will prefix any remote messages sent by USBMJOB.
`RMTMSGPREX`	Specifies the host name of the remote system on which the **`uwto`** command is executed.
`RMTHOST`	Specifies the path (including the executable name) to the remote message handler (**`uwto`**).
`MSGCMDPATH`	Specifies the user id for the remote system where the **`uwto`** command resides.
`RMTUSER`	Specifies the password for the user on the remote system where the **`uwto`** command resides.
`RMTPWD`	Specifies a time interval (in seconds). The time interval will control how long a remote reply message will remain on a remote console without being replied to before it will time out.

17.5.3 Universal Submit Job from Windows / UNIX to OS/400

Figure 17.3, below, illustrates the issuing of a command to the remote OS/400 as a parameter of the USBMJOB.

```
ucmd -c "usbmjob cmd(dspsyssts)" -i ohio -u usrid -w usrpwd
```

Figure 17.3 Universal Submit Job – Windows / UNIX to OS/400

In this example, USBMJOB is submitted to the server running on the host **ohio**.

(Also see the Universal Command 3.2.0 User Guide for an example of USBMJOB.)

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-c	Universal Command command option (**usbmjob**).
-i	Directs the command to a computer with a host name of **as400**.
-u	Specifies the remote user ID with which to execute the command.
-w	Specifies the password for the remote user ID.

Chapter 18
Universal Write-to-Operator

18.1 Overview

The Universal WTO (UWTO) utility is a command line utility for the z/OS UNIX System Services (USS) environment.

Universal WTO issues two types of messages to z/OS consoles:

1. Write-To-Operator (WTO) messages
2. Write-To-Operator-with-Reply (WTOR) messages

Note: UWTO became available for the z/OS USS environment with Universal Command 2.2.0, Level 12.

18.2 Usage

Universal WTO either:

- Writes a message to the z/OS console, and then ends (WTO).
- Writes a message to the z/OS console and waits for a requested reply (WTOR).

The type of message to be written (WTO or WTOR) is specified via the REPLY option. If WTOR is specified, the message is written to the console as a WTOR message and Universal WTO waits for a reply. The message reply is written to stdout.

18.2.1 Return Codes

The UWTO command ends with specific return codes indicating the success of the requested action.

Table 18.1, below, describes these return codes.

Return Code	Description
0	Process was successful.
1	Message was written to the console, but a warning was issued regarding a requested option. A detailed message is written to standard error.
2	WTOR request timed out waiting for an operator reply.
3	Error occurred when attempting to write the message. No message was written to the console.
10	Error in the command line options was detected. No processing occurred.

Table 18.1 Universal WTO - Return Codes

18.2.2 Configuration Options

Table 18.2, below, identifies the Universal WTO configuration options.

Each **Option Name** is a link to detailed information about that configuration option in the Universal Products Utilities 3.2.0 Reference Guide.

Option Name	Description
CONSOLE_ID	ID of the console to which to route the message.
CONSOLE_NAME	Name of the console to which to route the message.
HELP	Writes a description of the command options and their format.
KEY	Key to associate with the message.
MESSAGE	Text to write to the z/OS operator console.
MESSAGE_LEVEL	Level of messages to write.
REPLY	Directs UWTO is issue a WTOR message and wait for an operator reply to the message.
TIMEOUT	Number of seconds to wait for a WTOR operator reply.
VERSION	Writes the program version and copyright information.

Table 18.2 Universal WTO - Configuration Options

18.2.3 Command Line Syntax

Figure 18.1, below, illustrates the syntax – using the long form of command line options – of Universal WTO.

```
uwto
[-msg message]
[-reply {yes|no} [-timeout seconds] ]
[ -consoleid id | -consolename name ]
[-level {trace|audit|info|warn|error}]
[-key keyname]

uwto
{ -help | -version }
```

Figure 18.1 Universal WTO - Command Line Syntax

See Section 2.2.1 Configuration Methods for complete details on configuration methods and command input for Universal Products.

18.3 Examples of Universal Write-to-Operator

This section contains examples demonstrating the use of Universal Write-to-Operator.

The following list provides a link to each example.

- USS UWTO for z/OS Console
- USS UWTO for z/OS Console and Wait for Reply

18.3.1 USS UWTO for z/OS Console

Figure 18.2, below, illustrates the issuing of a WTO message to the z/OS console.

No reply is required.

```
uwto -msg "This message is written to the Console"
```

Figure 18.2 Universal WTO - Issue WTO to z/OS Console

The message text "**`This message is written to the Console`**" will be written to the default z/OS consoles.

SYSIN Options

The SYSIN option used in this example is:

Option	Description
`-msg`	Specifies the text to write to the z/OS operator console. The text is written as a single-line WTO or WTOR message.

18.3.2 USS UWTO for z/OS Console and Wait for Reply

Figure 18.3, illustrates the issuing of a WTOR message to the z/OS console.

A reply is required.

```
uwto -msg "This message is written to the Console" -reply yes -timeout 120
```

Figure 18.3 Universal WTO - Issue WTOR to z/OS Console

The message text "**This message is written to the Console**" will be written to the default z/OS consoles.

The process will wait 120 seconds for a required reply. If a reply is not received within this time, the WTOR message is deleted and Universal WTO ends with exit code 2. The reply length is limited to 119 characters. The reply is written to UWTO's standard output file.

Note: A valid operator reply to a WTOR message can be zero characters. In this case, nothing is written to stdout.

SYSIN Options

The SYSIN options used in this example are:

Option	Description
-msg	Specifies the text to write to the z/OS operator console. The text is written as a single-line WTO or WTOR message.
-reply	Directs Universal WTO to issue a WTOR message and wait for an operator reply to the message.
-timeout	Specifies the number of seconds to wait for a WTOR operator reply. If a reply is not received within this time, the WTOR message is deleted and UWTO ends with exit code 2. Default is 0 (wait indefinitely).

Appendix A
Customer Support

Stonebranch, Inc. provides customer support, via telephone and e-mail, for Universal Products Utilities and all Universal Products.

TELEPHONE

Customer support via telephone is available 24 hours per day, 7 days per week.

North America

(+1) 678 366-7887, extension 6

(+1) 877 366-7887, extension 6 [toll-free]

Europe

+49 (0) 700 5566 7887

E-MAIL

All Locations

support@stonebranch.com

Customer support contact via e-mail also can be made via the Stonebranch website:

www.stonebranch.com

INDEX

C

W

X

Z

950 North Point Parkway, Suite 200
Alpharetta, Georgia 30005
U.S.A.