# Universal Command

## User Guide

Universal Products

Version 3.2.0

ucmd-user-3204

# Universal Command

# User Guide

## Universal Products 3.2.0

| Document Name | Universal Command 3.2.0 User Guide | | | | |
|---|---|---|---|---|---|
| **Document ID** | **ucmd-user-3204** | | | | |
| **Products** | **z/OS** | **UNIX** | **Windows** | **OS/400** | **HP NonStop*** |
| Universal Command Manager | √ | √ | √ | √ | √ |
| Universal Command Server | √ | √ | √ | √ | √ |
| * Universal Command 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 Command 3.2.0 User Guide (ucmd-user-3204) September 8, 2009

- Moved Universal Command Manager examples for all operating systems to Appendix A Examples.
- Added examples for Universal Command Manager for z/OS.
- Added a note about use of the **auto** value for the UCMD Manager RESTART configuration option in the Requesting Restart sub-section of Section 2.6.2 Manager Fault Tolerance

### Changes for Universal Command 3.2.0 User Guide (ucmd-user-3203) July 29, 2009

#### Universal Command 3.2.0.1 for OS/400

- Modified document for upgrade from Universal Command 3.1.1 for OS/400 to Universal Command 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 Section 6.2.1 Universal Products for OS/400 Commands.
  - Added the following options in Section 6.2.4 Configuration Options of Chapter 6 Universal Command Manager for OS/400:
    - ACTIVITY_MONITORING
    - ASSIGN_PROCESS_TO_JOB

    - CERTIFICATE_REVOCATION_LIST
    - COMMENT
    - CONNECT_TIMEOUT
    - DNS_EXPAND
    - EVENT_GENERATION
    - EXIT_CODE_MAP
    - HOST_SELECTION
    - MFT_SAFE_MODE
    - PLF_DIRECTORY
- Modified the following information in Section 11.2.1 User Command Environment of Chapter 11 Universal Command Server for OS/400:
    - Initiator (UCMSINIT) Exit Points
    - User Command Exit Code
- Added the following options in Section 11.5.3 Configuration Options Summary of Chapter 11 Universal Command Server for OS/400:
    - ACTIVITY_MONITORING
    - EVENT_GENERATION
    - LOGIN
    - USE_USER_ACCOUNTING_CODE

## Changes for Universal Command 3.2.0 User Guide (ucmd-user-3202) April 1, 2009

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

## Changes for Universal Command 3.2.0 User Guide (ucmd-user-3201) September 5, 2008

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

## Changes for Universal Command 3.2.0 User Guide (ucmd-user-320) May 16, 2008

### Universal Command 3.2.0.0

- Added support for the following features:

    - Return Code Translation
      Return code conventions used on the operating system on which the Manager executes and the operating system on which the Server executes are often very different. Return code translation options now allow you to translate the user job's return code to a different return code more suitable for the Manager operating system.
    - Automatic Generation of Command Identifiers
      Manager command identifiers, used primarily in the Manager Fault Tolerant (MFT) feature, are used to uniquely identify the unit of work on the remote Broker. Unique command identifiers can now be automatically generated by the Manager. Automatic generation greatly simplifies command identifier specification in scheduling and automation systems.
    - Maximum Script Size Increase
      The size of the script that a Manager can send to the Server for execution has increased from approximately 64K to 6M.
- Added the following sections in Chapter 2 Features:
    - Section 2.4 Universal Configuration Manager
    - Section 2.7 z/OS CANCEL Command Support
- Added the following Universal Command Manager configuration options:
    - ACTIVITY_MONITORING
    - ASSIGN_PROCESS_TO_JOB
    - BIF_DIRECTORY
    - COMMENT
    - CONNECT_TIMEOUT
    - DNS_EXPAND
    - EVENT_GENERATION
    - EXIT_CODE_MAP
    - HOST_SELECTION
    - MFT_SAFE_MODE
    - PLF_DIRECTORY
    - SAF_KEY_RING
    - SAF_KEY_RING_LABEL
    - SERVER_STOP_CONDITIONS
    - SSL_IMPLEMENTATION
    - SYSTEM_ID
- Added the following Universal Command Server configuration options:
    - ACTIVITY_MONITORING
    - ASSIGN_PROCESS_TO_JOB
    - EVENT_GENERATION
    - SCRIPT_TYPE
- Added information for support of Certificate Revocation Lists (CRLs)

## Changes for Universal Command 3.1.1 User Guide (ucmd-user-31110) February 28, 2007

- Added customer support telephone number for Europe to Appendix B Customer Support.

### Universal Broker 3.1.1.8

- Added TMP_DIRECTORY configuration option in Section 3.2.3.2 Configuration Options Reference of Section 3.2 Universal Broker for z/OS.
- Added SERVICE_BACKLOG configuration option in the following sections:
  - Section 3.2.3.2 Configuration Options Reference of Section 3.2 Universal Broker for z/OS.
  - Section 3.3.2.2 Configuration Options Reference of Section 3.3 Universal Broker for Windows.
  - Section 3.4.2.2 Configuration Options Reference of Section 3.4 Universal Broker for UNIX.
- Deleted COMPONENT_PORT configuration option from the following sections:
  - Section 3.2.3.2 Configuration Options Reference of Section 3.2 Universal Broker for z/OS.
  - Section 3.4.2.2 Configuration Options Reference of Section 3.4 Universal Broker for UNIX
  - Section 3.5.3.2 Configuration Options Reference of Section 3.5 Universal Broker for OS/400
  - Section 3.6.2.2 Configuration Options Reference of Section 3.6 Universal Broker for HP NonStop

### Universal Command 3.1.1.7

- Added TMP_DIRECTORY configuration option in Section 4.2.4.4 Configuration Options Reference of Section 4.2 Universal Command Server for z/OS.
- Specified that the acceptable value must be greater than 0 for the STDIO_TIMEOUT option in the following sections:
  - Section 4.2.4.4 Configuration Options Reference of Section 4.2 Universal Command Server for z/OS
  - Section 4.3.4.3 Configuration Options Reference of Section 4.3 Universal Command Server for Windows
  - Section 4.4.4.4 Configuration Options Reference of Section 4.4 Universal Command Server for UNIX
  - Section 4.5.4.4 Configuration Options Reference of Section 4.5 Universal Command Server for OS/400
- Added time value to MESSAGE_LEVEL in Section 5.4.2.3 Command Options Reference of Section 5.4 Universal Command Manager for OS/400.

### Universal Control 3.1.1.7

- Added TMP_DIRECTORY configuration option in Section 6.2.3.2 Configuration Options Reference of Section 6.2 Universal Control Server for z/OS.

### Universal Submit Job 3.1.1.0

- Added default value for REMOTE REFRESH INTERVAL option in Section 7.11.2.6 Universal Submit Job (USBMJOB) Specific Options.

## Changes for Universal Command 3.1.1 User Guide (ucmd-user-3119) December 15, 2006

### Universal Broker 3.1.1.7

- Specified that Universal Broker job logs automatically are sent to a designated output queue and printer device in Section 3.5.1 Environment of Section 3.5 Universal Broker for OS/400.
- Specified that Universal Broker wraps the job log when it is full, rather than stopping the job, in Section 3.5.1 Environment of Section 3.5 Universal Broker for OS/400.

### Universal Command 3.1.1.6

- Added member fields to the following options in Figure 41. Universal Command for OS/400 Manager Syntax and Section 5.4.2.3 Command Options Reference of Section 5.4 Universal Command Manager for OS/400:
  - CA_CERTIFICATES
  - CERTIFICATE
  - PRIVATE_KEY

### Universal Submit Job 3.1.1.0

- Specified spool control (*SPLCTL) authority information in:
  - Section 7.11 Universal Submit Job
  - Section 3.5.5.3 User Profile of Section 3.5 Universal Broker for OS/400
  - Section 6.5.1.1 User Identification of Section 6.5 Universal Control Server for OS/400
  - User Identification in Section 4.5.2.1 User Command Environment of Section 4.5 Universal Command Server for OS/400

## Changes for UNV Release 3.1.1.8 November 15, 2006

- Added List of Figures and List of Tables.
- Added Customer Support.
- Added OS/400 entry to Section 2.5.1 UACL Configuration.
- Added the following OS/400 Universal Broker configuration options to Section 3.5.3.1 Configuration Options Summary and Section 3.5.3.2 Configuration Options Reference:
  - CA_CERTIFICATES

    - CERTIFICATE
    - COMPONENT_PORT
    - CTL_SSL_CIPHER_LIST
    - DNS_CACHE_TIMEOUT
    - PRIVATE_KEY
    - PRIVATE_KEY_PWD
    - TRACE_FILE_LINES
    - TRACE_TABLE
- Added the following OS/400 Universal Broker component definitions to Section 3.5.4.1 Component Definitions:
    - Component Name
    - Component Type
- Added Section 3.5.5.4 Universal Access Control List to Universal Broker for OS/400.
- Defined Universal Command Server Initiator program (UCMSINIT) in the Universal Command Server for OS/400 Section 4.5.2.1 User Command Environment.
- Documented how to execute a Single REXX Line in Universal Command Server for OS/400 Section 4.5.2.2 User Command.
- Documented Command References in Universal Command Server for OS/400 Section 4.5.2.3 Command References.
- Added the following OS/400 Universal Server configuration options to Section 4.5.4.3 Configuration Options Summary and Section 4.5.4.4 Configuration Options Reference:
    - ALLOW_SPOOLING
    - COMMAND_TYPE
    - DATA_SSL_CIPHER_LIST
    - JOB_RETENTION
    - NETWORK_FAULT_TOLERANT
    - STDIO_TIMEOUT
    - TRACE_FILE_LINES
    - TRACE_TABLE
- Added Section 4.5.5.4 Universal Access Control List to Universal Command Server for OS/400.
- Added Command Execution Environment to Section 5.4 Universal Command Manager for OS/400.
- Removed Command Line, Short Form and Command Line, Long Form from all command options in the Section 5.4 Universal Command Manager for OS/400 and Section 6.10 Universal Control Manager for OS/400.
- Removed "Command file" as a source for UCMD Manager for OS/400 command options in Section 5.4.2.1 Command Syntax of Section 5.4 Universal Command Manager for OS/400.
- Added command option categories to table in Section 5.4.2.2 Command Options Summary in Universal Command Manager for OS/400.
- Added additional configuration options to Section 5.4.2.2 Command Options Summary of Universal Command Manager for OS/400.

- Added Certificate Category Options Summary and Miscellaneous Category Options Summary to Section 5.4.2.2 Command Options Summary of the Universal Command Manager for OS/400.
- Added the following command options to Section 5.4.2.3 Command Options Reference of Universal Command Manager for OS/400.
  - CA_CERTIFICATES
  - CERTIFICATE
  - COMMAND_ID
  - COMMAND_TYPE
  - CTL_SSL_CIPHER_LIST
  - DATA_SSL_CIPHER_LIST
  - DEFAULT_CIPHER
  - FORCE_COMPLETE
  - HOSTNAME_RETRY_COUNT
  - JOB_RETENTION
  - MANAGER_FAULT_TOLERANT
  - NETWORK_DELAY
  - NETWORK_FAULT_TOLERANT
  - OUTBOUND_IP
  - PRIVATE_KEY
  - PRIVATE_KEY_PWD
  - RECONNECT_RETRY_COUNT
  - RECONNECT_RETRY_INTERVAL
  - RESTART
  - SCRIPT_TYPE
  - TRACE_FILE_LINES
  - TRACE_TABLE
  - VERIFY_HOST_NAME
  - VERIFY_SERIAL_NUMBER
  - VERSION
- Modified the following command options in Section 5.4.2.3 Command Options Reference of Universal Command Manager for OS/400:
  - DATA_COMPRESSION
  - MESSAGE_LEVEL
  - NETWORK_ROLE
  - SERVER_OPTIONS
- Deleted the following command options from Section 5.4.2.3 Command Options Reference of Section 5.4 Universal Command Manager for OS/400:
  - SIO_DATA_AUTHENTICATION
  - SIO_DATA_COMPRESSION
  - SIO_DATA_ENCRYPTION
  - STDIN_FILE_SPEC
  - STDERR_FILE_SPEC
  - STDOUT_FILE_SPEC

- Deleted requirement for spacing before the first option name in SERVER_OPTIONS command option in Section 5.4.2.3 Command Options Reference of Section 5.4 Universal Command Manager for OS/400.
- Added the following to Section 5.4.2.4 Examples of Universal Command Manager for OS/400.
  - File Copy Example 3
  - File Copy Example 4
- Deleted HOSTNAME_RETRY_COUNT, FORCE_COMPLETE, and RESTART from the options summary tables in Section 5.5 Universal Command Manager for HP NonStop.
- Added Section 6.5 Universal Control Server for OS/400.
- Added Section 6.10 Universal Control Manager for OS/400.
- Added additional options to Figure 80. STRUCP Command Syntax.
- Added the following command options to Section 7.2.4.2Command Options Reference of Universal Copy for OS/400.
  - INPUTMODE
  - OUTPUTMODE
  - VERSION
- Modified the following command options in Section 7.2.4.2 Command Options Reference of Universal Copy for OS/400.
  - MSGLEVEL
- Modified Figure 90  Universal Encrypt Translator Syntax.
- Added the following command options to Section 7.3.5.2 Command Options Reference for Universal Encryption for OS/400:
  - CODE_PAGE
  - VERSION
- Added Section 7.3.5.3 Example to Universal Encryption for OS/400.
- Modified the command syntax for Universal Message Translator for OS/400 in Section 7.5.5.2 Command Syntax.
- Added Section 7.9.5 OS/400 for Universal Query utility.
- Specified that Universal Command 1.2.1 is required for Universal Submit Job utility in Section 7.11.2.2 Return Codes.
- Specified that the Universal Display Log File utility is available with Universal Command 1.2.1 Level 10 in Section 7.12 Universal Display Log File.
- Added illustration for command syntax in Section 7.12.2.1 Command Syntax, and added Section 7.12.2.2 Command Options Reference header, for Universal Display Log File utility.
- Changed -localcodepage to -codepage in CODE_PAGE and SIO_LOCAL_CODE_PAGE in sub-section 5.4.2.3 Command Options Reference of Section 5.4 Universal Command Manager for OS/400.
- Added Command Syntax Rules to sub-section 7.11.2.4 Command Syntax in Section 7.11 Universal Submit Job.
- Added REMOTE REPLY PORT command option to Section 7.11.2.6 Universal Submit Job (USBMJOB) Specific Options.
- Added 7.3.5.3 Example to sub-section 7.3.5 OS/400 Universal Encrypt.

- Added AES to Figure 90 Universal Encrypt Translator Syntax.
- Added AES to Section 7.3.5.2 Command Options Reference.
- Modified description of ENCRYPTION_KEY in Section 7.3.5.2 Command Options Reference.

### Changes for Release 3.1.1
### January 30, 2006

- Added AES 256-bit CBC mode encryption to UENCRYPT.
- Added support for UENCRYPT'ed AES command files to Universal Command, Universal Certificate and Universal Control.

### Changes for Release 3.1.1
### April 30, 2005

- Addition of the Universal Certificate utility to create X.509 certificates.
- Addition of parameter OUTBOUND_IP that enables the specification of a particular IP address the managers bind to for outgoing TCP/IP connections.
- Addition of the Broker parameter UCMD_STC_SUPPORT that specifies whether or not the Broker establishes the environment (SMF exit, etc.) to support Universal Command start task execution.
- Parameter STDIO_TIMEOUT was added to specify the amount of time in seconds that the Universal Command Server process will wait for standard I/O to be closed by child processes after the parent process has completed. Once this time has expired, the server process will exit.
- Addition of parameter CODEPAGE to the Universal Encrypt utility program.

### Changes for Release 3.1.0
### October 31, 2004

- Addition of SSL and X.509 certificates for the network protocol and peer authentication.
- The Universal Command Server DATA_ENCRYPTION option does not force data encryption for SSL sessions as it does for Universal Command version 2 sessions. The Server DATA_SSL_CIPHER_LIST option controls whether data encryption is required or not for SSL sessions.
- Addition of the Command Reference feature.
- Changing the MVS Universal Broker and Universal Command Server from the UNIX System Services (USS) environment to the native MVS environment.
- Addition of MVS Universal Command Server support for the execution of Started Task address spaces.
- New Universal Access Control List rules based on request type.
- Addition of X.509 certificate integration with Universal Access Control Lists.

- New Universal Command return codes. The return code 1001 is no longer used on MVS and Windows.
- Improved Universal Broker performance using DNS caching.

### Changes for AS/400 Release 1.2.1
### February 12, 2004

- Enhanced documentation for the USBMJOB command. The enhancements include added documentation for the new CPYSPLF command parameter.
- Addition of the UDSPLOGF command for the AS/400.

### Changes for HP NonStop Release 2.1.1
### February 3, 2004

- Minor documentation changes in the HP NonStop Broker log file names and the HP NonStop ucopy program options. Changes are related to maintenance level 2 of HP NonStop 2.1.1.

### Documentation Change
### November 24, 2003

- Moved installation instructions for all platforms to a separate document, the Universal Products Installation Guide.

### Changes for HP NonStop Release 2.1.1
### September 29, 2003

- Addition of the TZ environment variable to the HP NonStop ubrokerd TACL script to set the appropriate time zones and offset.
- Addition of the -script_type Server side option to allow a user to specify whether they are executing TACL or OSS user commands.
- Addition of the -cpu Server side option to allow the user to specify the processor that the user command is to be executed on.
- Addition of the -priority Server side option to allow the user the specify the priority of the user command being executed.

### Changes for Release 2.2.0
### August 8, 2003

- Addition of the TZ environment variable to the z/OS Universal Command JCL procedure in order to set the appropriate time zones and offset.

- Addition of pattern matching capabilities, case-sensitivity switch and variable names to the Universal Access Control List rules.

### Changes for HP NonStop Release 2.1.1
### July 31, 2003

- Addition of HP NonStop NSK G06 2.1.1.

### Changes for Release 2.2.0
### May 12, 2003

- Addition of the Universal WTO command for z/OS UNIX System Services.
- Addition of the Universal Submit Job command for the AS/400.

### Changes for Release 2.2.0
### February 7, 2003

- General improvements to the Windows installation chapter.
- Addition of the FORCE_COMPLETE Universal Command Manager option.
- Addition of a common UNIX installation script for AIX, HP-UX and Solaris that executes the vendor specific installation methods.
- Addition of the MANAGER_FAULT_TOLERANT option to specify whether the manager fault tolerant feature is used or not. This option maintains backward compatibility with the COMMAND_ID option in Universal Command releases prior to 2.2.0's and changes the behavior of the COMMAND_ID option in earlier 2.2.0 releases. As of this release, the COMMAND_ID does not turn on the manager fault tolerant feature. Manager fault tolerance is controlled with the MANAGER_FAULT_TOLERANT option.

### Changes for 2.2.0
### November 18, 2002

- Availability of Windows NT, Windows 2000, and Windows XP ports.
- Addition of the z/OS Manager standard input option SIO_TRAILING_SPACES to preserve all bytes of a fixed format record.
- Changed the Host Fault Tolerant feature name to Manager Fault Tolerant.

### Changes for Release 2.2.0
### October 11, 2002

- Added the Document Structure section describing conventions and terminology used throughout the document.

- Miscellaneous formatting corrections.

### Changes for Release 2.2.0
### September 17, 2002

- Host Fault Tolerant feature to permit restarts of terminated managers.
- Universal Command spooling system to save standard I/O files.
- Addition of the LOGIN option to invoke users shell environment under which user commands will execute.
- Addition of circular trace files to help manage disk space utilization.
- Universal Products installed using vendor packaging methods for AIX, HP-UX, and Solaris operating systems.
- Universal Access Control Lists to secure access to Universal Products services.
- Addition of SCRIPT_TYPE option to invoke Windows scripts other then batch files.
- Universal Copy enhancements for z/OS and transactional copies.
- Addition of HOSTNAME_RETRY_COUNT to retry failed host name resolution attempts.

### Changes for Release 2.1.0
### April 22, 2002

- Addition of the compression method option, providing the ability to specify either the ZLIB or HASP compression algorithm for network data transfers.

### Changes for 2.1.0
### January 3, 2002

- Addition of the text translation method option as a performance enhancement for large text transfers.

### Changes for Release 2.1.0
### November 1, 2001

- Addition of Universal Command network fault tolerant option. Network Fault Tolerance enables Universal Command Manager and Server to continue operations when network faults are detected.
- Addition of Universal Query 2.1.0. Universal Query provides a user interface into the status of Universal Broker and its components.
- Addition of Universal Control 2.1.0. Universal Control provides a method to stop and control Universal components running under the control of Universal Broker.
- Addition of HP Trust System authentication.

- Addition of AUDIT message level option. The AUDIT level prints additional messages that provide more details of a specific execution of Universal Manager or Server.
- Removal of the NETWORK ROLE option. The network role is always active now.
- Addition of ACF2 configuration documentation.
- Changed encryption key from 56-bit to 128-bit.
- Changed Universal Command OS/390 Manager's TCP/IP stack from the MVS stack to the UNIX System Services stack.
- Universal Encrypt 2.1.0.
- Updated Universal Command NT Server to load the profile of the user account used to execute the requested process. This causes that user's registry hive to be mapped to HKEY_CURRENT_USER for the process.

### Changes for Release 1.2.0
### May 14, 2001

- General documentation improvements.

### Changes for Release 1.2.0
### April 15, 2001

- Addition of OS/400 1.2.1.

### Changes for Release 1.2.0
### February 7, 2001

- Changed product distribution CD from Juliet format to ISO 9660 format.

### Changes for Release 1.2.0
### January, 2001

- Addition of a network keep-alive option. A network keep-alive permits early detection of network failures and prevents connection timeouts in firewalls for commands that execute for a long period of time without any data exchange across the network.
- Updated UNIX installation.
- Improved network data compression. The new compression implementation is a 50% improvement in speed and 30% improvement in the compression ratio on average.
- New single port connection protocol. The Manager connects to the Broker's static port (7887) only. Single port connections permit network firewall administrators to easily permit Universal Command through firewalls.
- Addition of an OS/390 RACF interface for Universal Command Manager. RACF profiles can be defined to control access to remote systems as specific users via Universal Command Manager on OS/390.

- Addition of encrypted command files. Universal Command Manager command options can be encrypted using the new program uencrypt. The Manager can than be instructed to read its options from the encrypted command file. This permits encryption of user identifiers and passwords.
- Addition of Universal Message Translation (UMET). UMET translates messages into an exit code. UMET will translate the error messages from commands into an exit code that can be used by batch schedulers and automation routines to determine success or failure of a command.
- Addition of a Messages and Codes manual.

### Changes for Release 1.1.0<br>November, 2000 Release

- Additions for AS/400 Universal Broker, Universal Command Server and Universal Command Manager. Because of the additions, chapter numbers changed from chapter 7 forward.

# 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, the Universal Command 3.2.0 User Guide has been reformatted and restructured.

Most importantly, links to detailed information in a companion document, the Universal Command 3.2.0 Reference Guide, have been created in this 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).

Additionally, information on Universal Broker and Universal Control now is documented in separate user guides / reference guides.

# Conventions

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

## 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

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

**Stoneman's Tip**

## 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:

- Universal Command Overview (Chapter 1)
  General architectural and functional overview of Universal Command.
- Features (Chapter 2)
  Description of Universal Command features, including configuration methods, network protocols, and command references.
- Universal Command Manager for z/OS (Chapter 3)
  Description of Universal Command Manager specific to the z/OS operating system.
- Universal Command Manager for Windows (Chapter 4)
  Description of Universal Command Manager specific to the Windows operating system.
- Universal Command Manager for UNIX (Chapter 5)
  Description of Universal Command Manager specific to the UNIX operating system.
- Universal Command Manager for OS/400 (Chapter 6)
  Description of Universal Command Manager specific to the OS/400 operating system.
- Universal Command Manager for HP NonStop (Chapter 7)
  Description of Universal Command Manager specific to the HP NonStop operating system.
- Universal Command Server for z/OS (Chapter 8)
  Description of Universal Command Server specific to the z/OS operating system.
- Universal Command Server for Windows (Chapter 9)
  Description of Universal Command Server specific to the Windows operating system.
- Universal Command Server for UNIX (Chapter 10)
  Description of Universal Command Server specific to the UNIX operating system.
- Universal Command Server for OS/400 (Chapter 11)
  Description of Universal Command Server specific to the OS/400 operating system.
- Universal Command Server for HP NonStop (Chapter 12)
  Description of Universal Command Server specific to the HP NonStop operating system.
- User Scripts (Chapter 13)
  Sample user scripts.
- Examples (Appendix A)
  Operating system-specific examples that demonstrate the use of Universal Command.
- Customer Support (Appendix B)
  Customer support contact information for Universal Command (and all Universal Products).

# Chapter 1
# Universal Command Overview

## 1.1 Overview

This chapter provides general information on Universal Command and is intended for the first time user. Detailed product information is provided in the following sections of this document.

# 1.2 What is Universal Command?

One of the most fundamental functions of an operating system is to provide a means for user interaction. User interfaces take many forms; the most common types are command line and graphical. The command line interface is provided by virtually all general-purpose operating systems today.

Universal Command (UCMD) provides a command line interface that extends to any operating system that can be reached on the computer network. That is, the remote operating system's command line interface is extended to the local operating system's command line interface. The remote and local systems can be running two different operating systems. The Universal Command interface is operating-system independent.

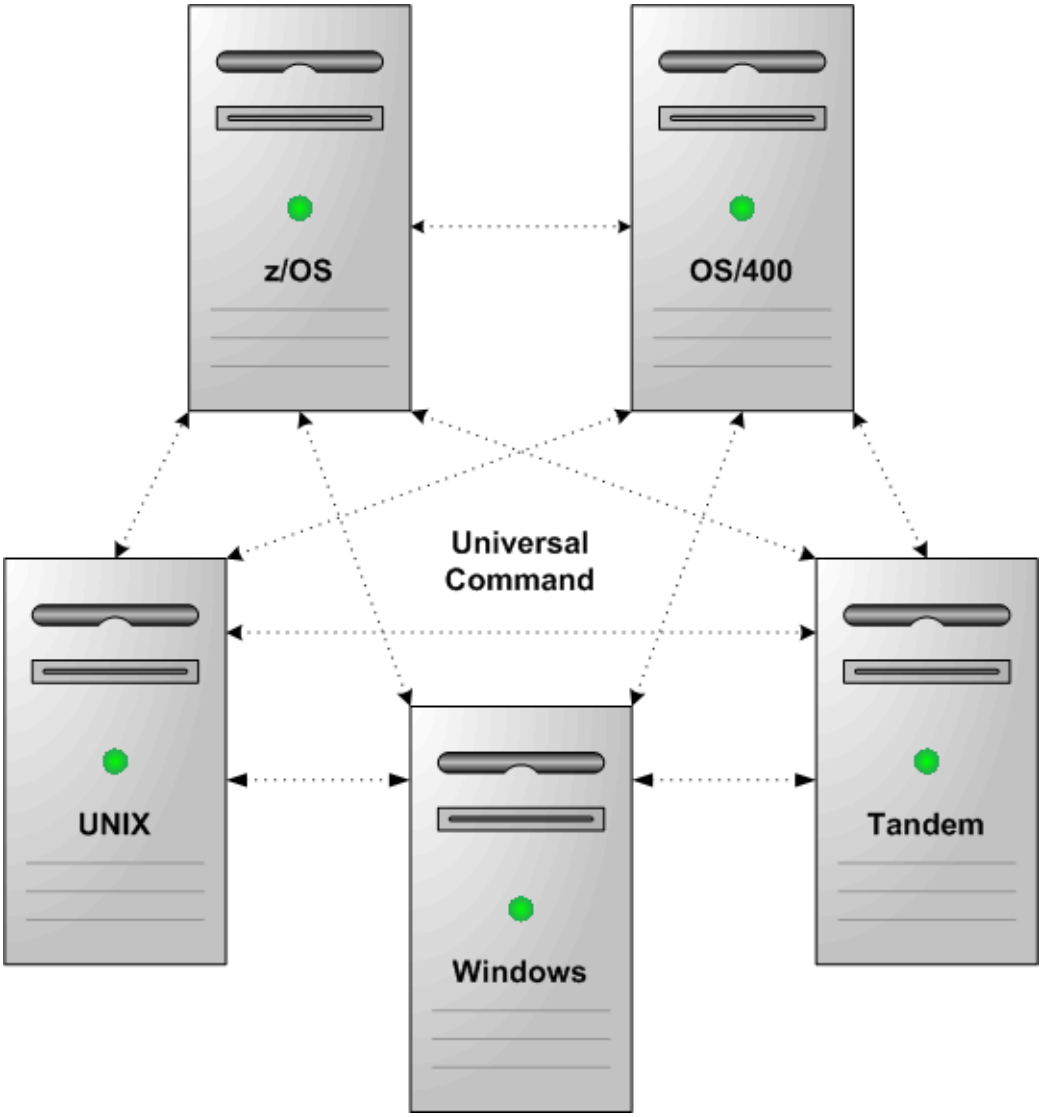


Figure 1.1 Universal Command Operating Systems Interface

# 1.3 What Can Universal Command Do for Me?

Businesses learn more ways every day to leverage technology for a competitive advantage. The Information Technology (IT) infrastructure consists of a diverse array of software and hardware systems. Database management, transaction management, resource planning, information warehouse, customer support, e-mail, web servers, and much more are required to sustain a business's technological advantage. This array of corporate software runs on a large variety of hardware platforms, which in turn run a variety of operating systems. The management of such technology grows more complex each year, if not each month.

The methods, processes and personnel used to manage the computing environment are as much a part of the business's technology investment as is the software and hardware being managed. Replacing or altering these proven management techniques and tools can be costly as well as risky to a business's success.

Universal Command leverages the management resources of today to manage the technology of tomorrow. For example, the z/OS computing environment has been centered around the batch process for years, and for good reason. Nothing else has proven itself to be more easily and reliably managed. Universal Command now permits the management of distributed platforms, such as UNIX and Windows, using the same reliable z/OS batch process. The batch processes used to forecast, schedule, manage output, manage archives, etc., can be used to manage the distributed platforms in the same exact manner.

# 1.4 How Does It Work?

The Universal Command product consists primarily of two components:

1. Universal Command Manager runs on the local system. It provides command line interface to a remote system.
2. Universal Command Server runs on the remote systems. It executes commands on behalf of Universal Command Managers.

The Manager supplies input files to, and receives output files from, the remote command in real-time. All files with character data are translated to the appropriate code pages for the respective system. The transmitted data, optionally, can be compressed, encrypted, or authenticated.

The Manager runs as long as the remote command is running. When the remote command ends, the Manager ends. The exit code of the remote command is used as the exit code of the Manager.

Any type of program, command, or script file that can be run from the command line interface can be run by Universal Command. It's that easy!

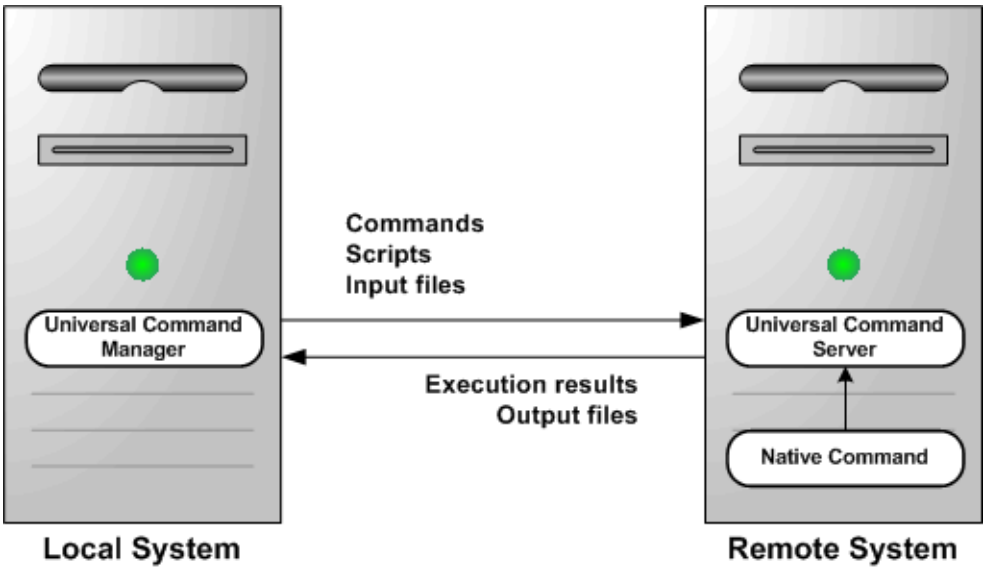


Figure 1.2 Universal Command Manager and Server

.

# Chapter 2
# Features

## 2.1 Overview

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

- Configuration
- Universal Configuration Manager
- Remote Configuration
- Network Data Transmission
- Fault Tolerance
- z/OS CANCEL Command Support
- Universal Access Control List
- Message and Audit Facilities
- Command References
- 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.

Configuration can be done either by:

- Setting default options and preferences for all executions of Universal Command.
- Setting options and preferences for a single execution of Universal Command.

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 to 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 in z/OS 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 UCMDLEVEL to a value of INFO:

```
PARM='ENVAR("UCMDLEVEL=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 UCMDLEVEL to a value of INFO in a bourne, bash, or korn shell:

```
UCMDLEVEL=INFO
export UCMDLEVEL
```

### 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 UCMDLEVEL to a value of INFO:

```
SET UCMDLEVEL=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 UCMDLEVEL to a value of INFO:

```
ADDENVVAR ENVVAR(UCMDLEVEL) 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 UCMDLEVEL to a value of INFO:

```
PARAM UCMDLEVEL 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.)

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), which 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 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.

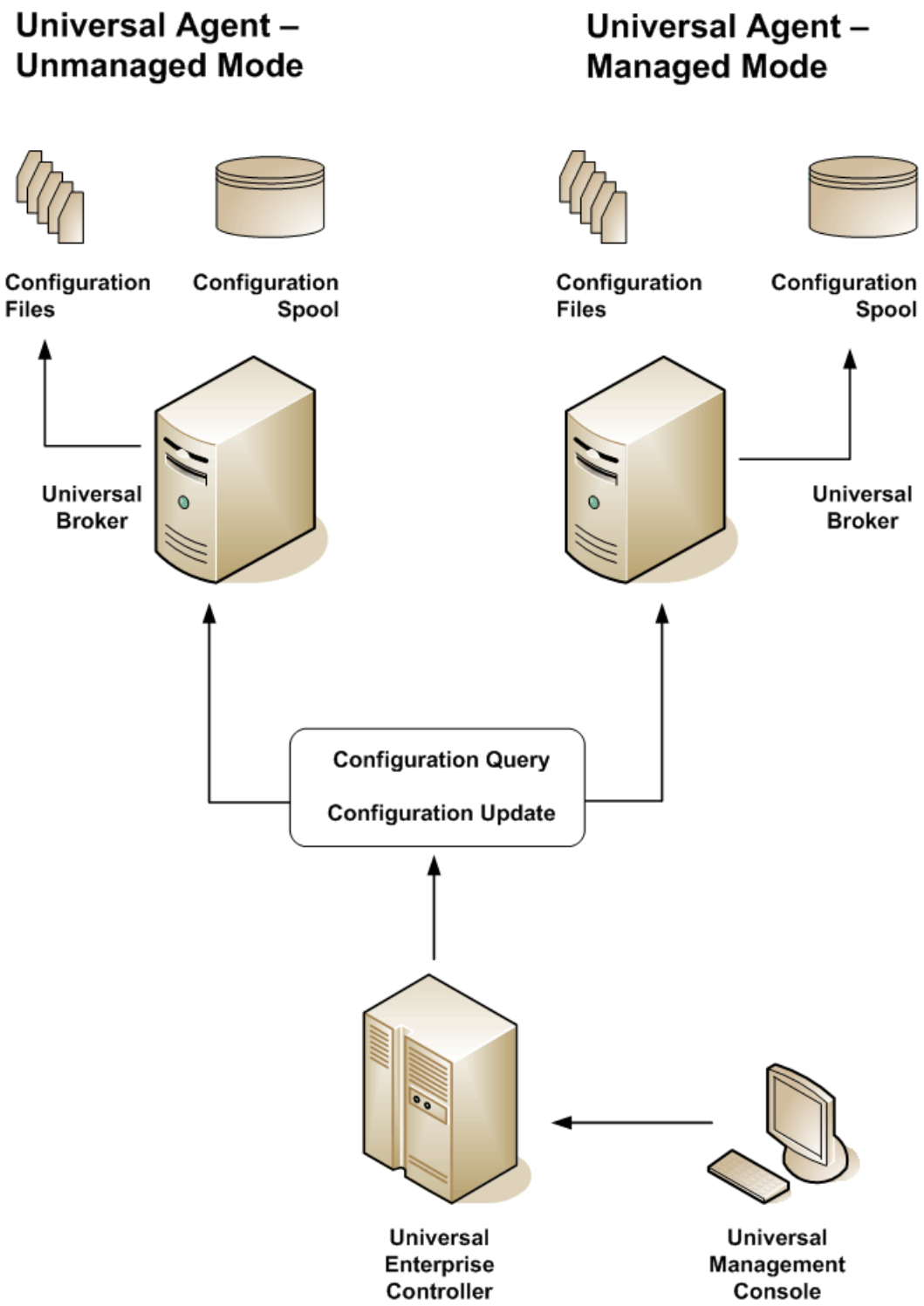


Figure 2.1 Remote Configuration - Unmanaged and Managed Modes of Operation

## 2.3.3 Universal Broker Startup

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

## 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:

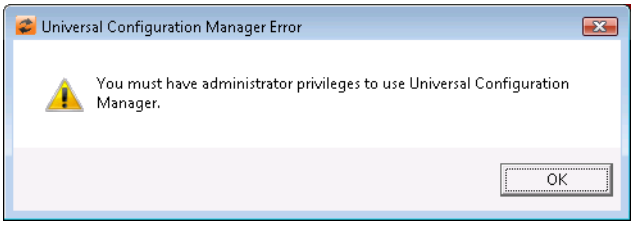


Figure 2.2 Universal Configuration Manager Error dialog - Windows Vista

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

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

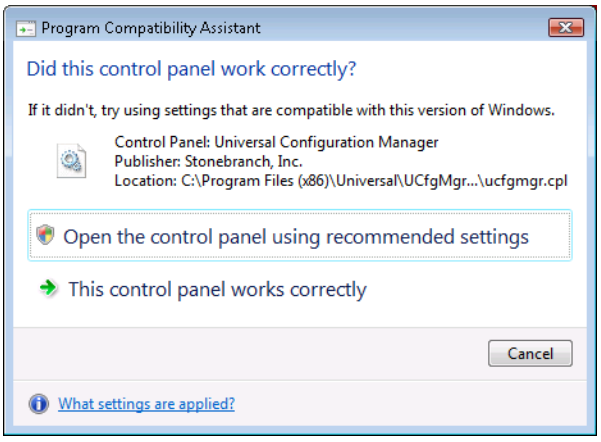


Figure 2.3 Windows Vista - Program Compatibility Assistant

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.

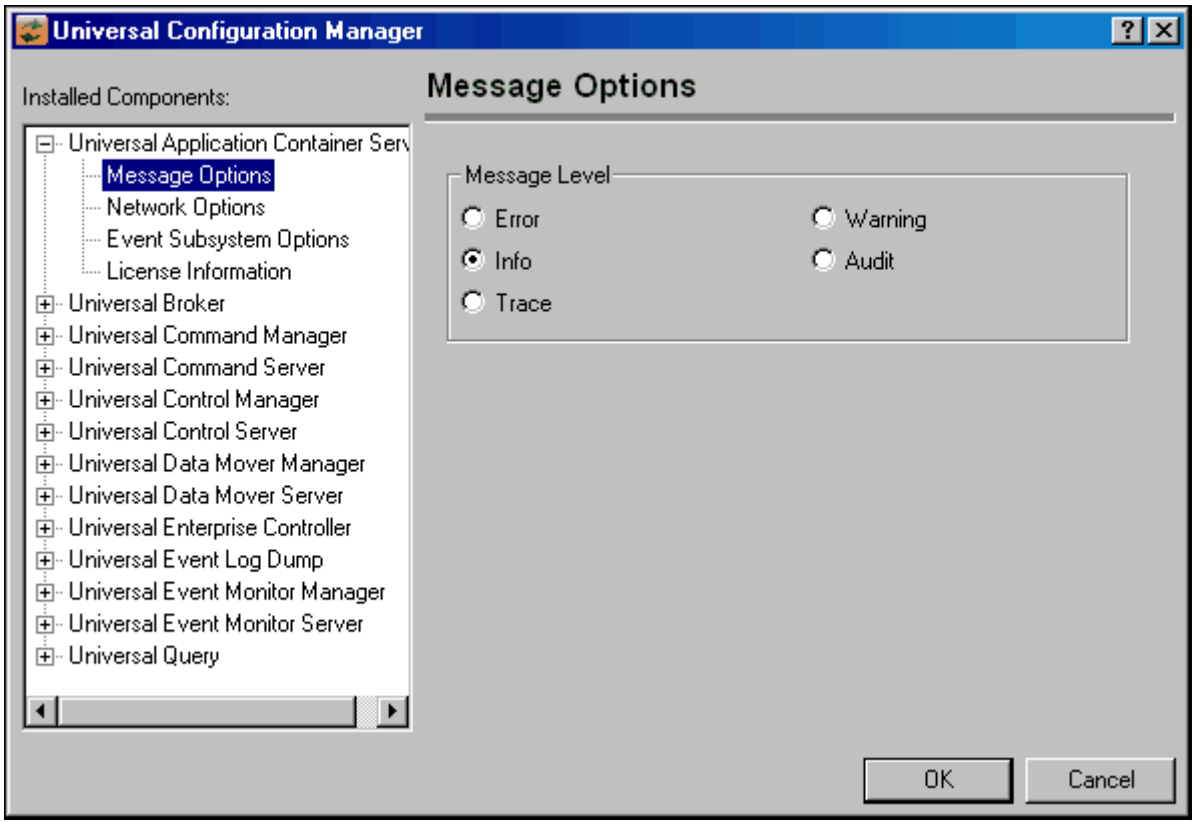


Figure 2.4 Universal Configuration Manager

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 **+** icons 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 in the configuration file, and Universal Broker is automatically 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 Command Installed Components

## Universal Command Manager

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

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

The text describes the selected component, Universal Command Manager.

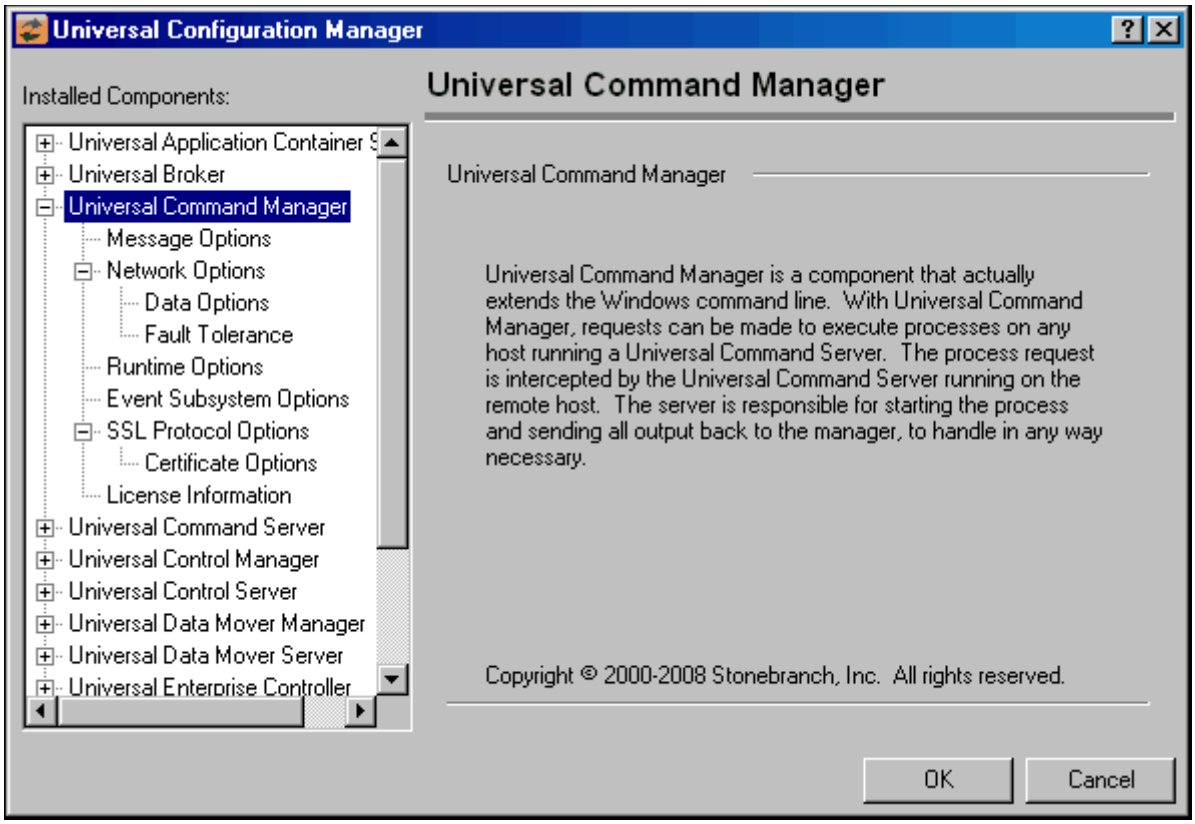


Figure 2.5 Universal Configuration Manager - UCMD Manager

## Universal Command Server

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

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

The text describes the selected component, Universal Command Server.

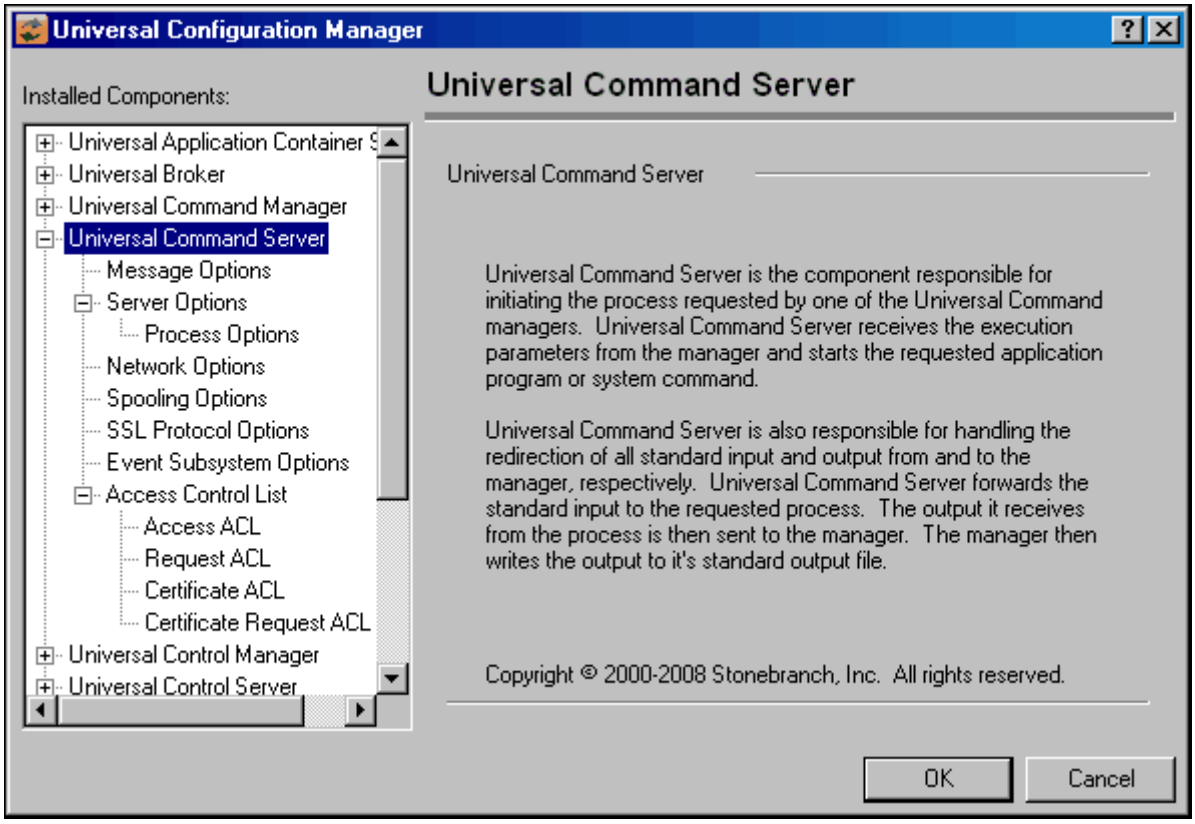


Figure 2.6 Universal Configuration Manager - UCMD Server

# 2.5 Network Data Transmission

Distributed systems, such as Universal Command, communicate over data networks. All Stonebranch 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 insure 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 2.11 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 insured 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 may be enabled and disabled on an individual bases.

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 affect 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 Fault Tolerance

Fault tolerant features address Universal Product capabilities to recover or restart from an array of error conditions that occur in any large IT organization. Errors occur as a result of human, software, or hardware conditions. The more resilient a product is to errors, the greater value it offers.

## 2.6.1 Network Fault Tolerance

Universal Command uses the TCP/IP protocol for communications over a data network. The TCP/IP protocol is a mature, robust protocol capable of resending packets and rerouting packets when network errors occur. However, data networks do have problems significant enough to prevent the TCP/IP protocol from recovering. As a result, the TCP/IP protocol terminates the connection between the application programs.

As with any application using TCP/IP, Universal Command is subject to these network errors. Should they occur, a product can no longer communicate and must shut down or restart. These types of errors normally show themselves as premature closes, connection resets, time-outs, or broken pipe errors.

Universal Command provides the ability to circumvent these types of errors with its Network Fault Tolerant protocol. By using this protocol, Universal Command traps the connection termination caused by the network error and reestablishes the network connections. When connections have been reestablished, processing resumes automatically from the location of the last successful message exchange. No program restarts are required and no data is lost.

The Network Fault Tolerant protocol acknowledges successfully received messages and checkpoints successfully sent messages. This reduces data throughput. Consequentially, the use of network fault tolerance should be weighed carefully in terms of increased execution time versus the probability of network errors and cost of such errors. For example, it may be easier to restart a program then to incur increased execution time.

When a network connection terminates, the manager enters a network reconnect phase. In this phase, the manager attempts to connect to the server and reestablish its network connections. The condition that caused the network error can persist only for seconds, or it can persist for days.

The manager attempts server reconnection for a limited amount of time, as specified by the following configuration options:

- RECONNECT_RETRY_COUNT (number of retry attempts)
- RECONNECT_RETRY_INTERVAL (frequency of retry attempts)

If all attempts fail, the manager ends with an error.

When a network connection terminates, the server's action depends on whether or not it is executing with manager fault tolerance.

Without manager fault tolerance, the server enters a disconnected state and waits for the manager to reconnect. The user process continues running. However, if the user process attempts any I/O on the standard files, it will block. The server waits for the manager to reconnect for a period of time defined by the manager's RECONNECT_RETRY_COUNT and RECONNECT_RETRY_INTERVAL. When that time has expired, the server terminates the user process and exits.

With manager fault tolerance, the server continues executing in a disconnected state. The server satisfies all user process standard I/O requests. The user process does not block. It continues to execute normally. When the user process ends, the server waits for a manager reconnect for a period of time defined by the JOB_RETENTION option.

## Universal Command Manager

Universal Command Manager starting with version 2.1.0 can request the use of the network fault tolerant protocol. If the server does not support the protocol or is not configured to accept the protocol, the manager continues without using the protocol.

The NETWORK_FAULT_TOLERANT option is used to request the protocol.

## Universal Command Server

Universal Command Server starting with version 2.1.0 can be configured with or without the network fault tolerant protocol. The NETWORK_FAULT_TOLERANT option is used to configure the network fault tolerant protocol.

If the server is configured with network fault tolerance off, the manager cannot override it. If the server is configured with the network fault tolerance on, the manager option determines if the protocol is actually used or not.

## 2.6.2 Manager Fault Tolerance

Distributed applications are comprised of many independent components running on host systems, throughout the enterprise, connected with a data network. Many of the host systems are in different physical locations, in different organization groups, and have different system management policies. It can be difficult to schedule individual host downtime when there are so many overlapping requirements.

Host systems must be shut down at scheduled intervals and, unfortunately, at unscheduled intervals. The impact of a system being down must be minimized by a distributed application.

With the Manager Fault Tolerant feature, Universal Command components can be shut down and restarted at a later time. After a manager has been started, it can be terminated and restarted at a later time. It can be shut down for any period of time: seconds, days, or even months. The server can be shut down as well. When it is restarted, any work that was completed prior to the shutdown is immediately available for a manager to retrieve.

### Functionality

The basic functionality of manager fault tolerance is:

1. Manager requests the execution of a command on a remote system.
2. Command executes on the remote system, optionally reading and writing data.
3. Manager redirects:
   - Its standard input data to the standard input of the remote command.
   - Standard output file and standard error file of the remote command to its own standard output and standard error.

If the manager is terminated or the manager's host system is shut down, the remote command cannot read the manager’s standard input or write its standard output and error files. Without manager fault tolerance, the remote command must terminate, since its data source and destination are now gone. Otherwise, it would wait forever.

Manager fault tolerance provides an execution environment in which the manager is not required in order for the user process to continue execution on the remote system. The user process can execute to completion with or without a manager connected.

When the manager starts a user process, the manager executes as normal; standard output and standard error files are redirected back to the manager as the user process produces the data. The difference is data spooling. In order for the user process to have real-time access to its input and output, the data is spooled in the Universal Spool Database. The spool provides complete independence from the manager. The spool subsystem satisfies all data requirements for the user process via the Universal Command Server.

The manager can terminate and a new manager can restart and reconnect to the user process. If the user process has completed, the new manager receives the user processes standard files and its exit status. The restarted manager behaves in all ways as if it was the originating manager.

### Command Identifier

A manager requests manager fault tolerance with the MANAGER_FAULT_TOLERANT option and by providing a command identifier (command ID) using the COMMAND_ID option. The command ID identifies the unit of work being executed. In this context, a unit of work includes the manager, server, and user process.

The manager indicates to the server that this request is restartable. The COMMAND_ID option provides a command identifier that uniquely identifies the server and user process on the remote host. When a manager is restarted, it must provide the same command ID identifying the server and user process with which it wants to reconnect.

Providing a unique command ID is not trivial. Many managers may be executing on many different hosts, and all executing work on the same server host. It is possible for a manager to start a restartable command from one host, terminate, and restart on a completely different host.

The command ID value can be any text value of unrestricted length. In practical terms, the character set and limits on command line length of the manager host impose restrictions on the value.

### Standard I/O Files

The Universal Spool system satisfies all user process data requests via the Universal Command Server. When the user process reads from its standard input file, the server reads it from the spool and provides it to the user process. When the user process writes to standard output or error, the server receives the data and writes it to the spool.

A manager requesting restart capability (manager fault tolerance) first transfers its entire standard input file to the server, which it in turns writes to the spool. When all data has been received, the server creates the user process. This provides complete manager independence for the entire life of the user process.

As long as the manager is connected, the standard output and standard error files are transferred to the manager, as the user process produces the data, all in real-time. The data also is written to the spool. If the manager terminates, the data is written to the spool only.

A restarted manager is sent all of the standard output and standard error files, from the beginning, that currently is spooled. If the user process still is executing, the restarted manager will receive all of the data currently spooled. When it has caught up with the data being produced, the manager starts to receive the data from the user process as it is written.

### Requesting Restart

When a restartable manager is initiated, it is either an initial instance or a restarted instance of a command ID. The command ID identifies a unit of work represented by the manager, server, and user process. See Command Identifier, above, for more information on the command ID.

The RESTART option specifies whether or not the manager instance is requesting a restart of a previous command ID. Possible RESTART values are **yes**, **no**, and **auto**.

The **auto** value specifies:

- If there is no existing command ID executing on the remote host, consider this manager execution the first instance.
- If there is an existing command ID, and it is not connected to any manager, consider this a restart of the command ID.

The **auto** value permits automatic restart by eliminating the need to modify the RESTART value for the initial instance and restarted instance.

Note: The **auto** value cannot be used with a COMMAND_ID value of *****, which specifies that the UCMD Manager will generate a unique command ID for each run.

## Case Example 1: Normal Execution

The following figure diagrams the sequence of events that occur when a restartable manager requests the execution of a command on a remote host. In this case the manager and server remain executing and connected until normal completion of the user process.

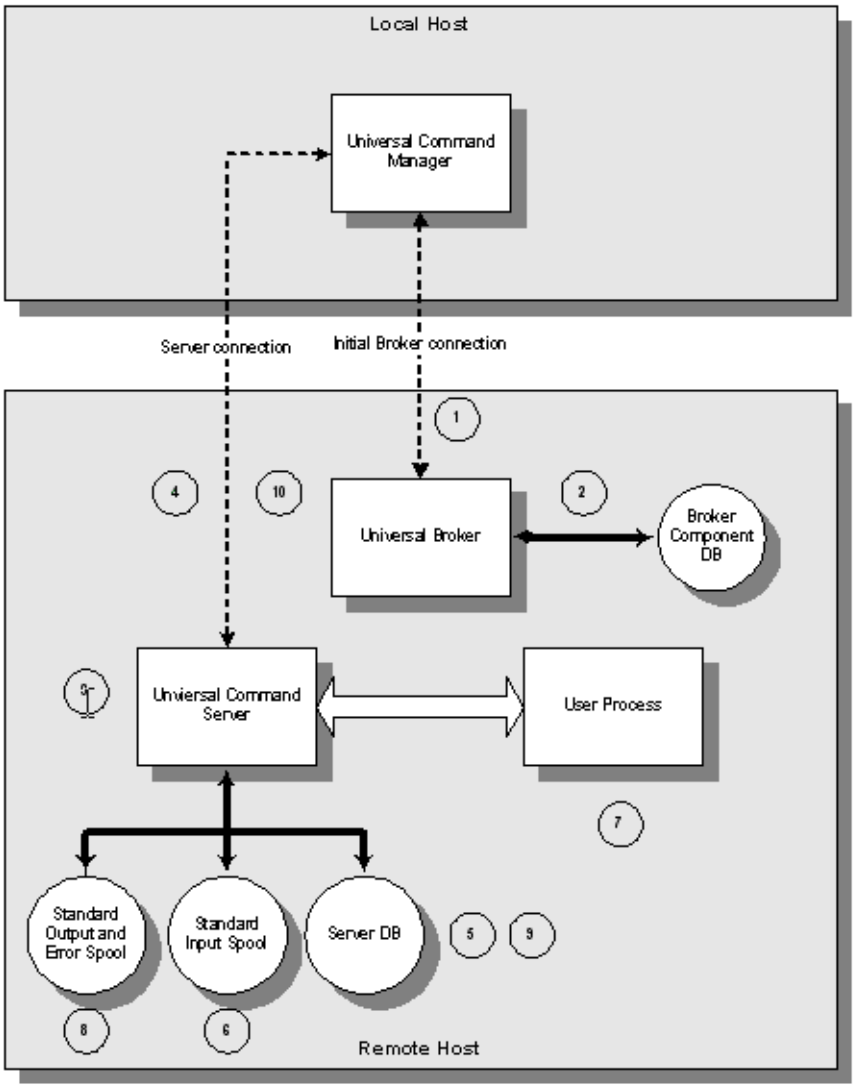


Figure 2.7 Case Example 1

The Local Host is the host on which the manager is being executed. The Remote Host is the host on which the manager is requesting command execution.

The components involved are:

- Universal Command Manager
  The manager requests remote execution of a command or script. The manager executes the remote command in a manner such that the command appears to be executed locally.
- Universal Broker
  The broker manages Universal component execution.
- Universal Command Server
  The server executes the manager requested command and processes the user process's standard I/O requests.
- User Process
  The user process represents the manager requested command.

The diagram demonstrates the sequence of events that occur when a restartable manager requests command execution on a remote host. The numbers enclosed in circles represent the sequence of events and correspond to the listed descriptions below.

1. The manager connects to the broker and sends a request to start a Universal Command Server. The start request from the manager requests manager fault tolerance and includes the command ID to identify the unit of work.
2. The broker records the unit of work in the Broker Component Database as restartable for possible future restarts.
3. The broker starts an instance of the Universal Command Server.
4. The manager and server exchange messages that specify all options used to carry out the request.
5. The server records the unit of work in the Universal Command Server Database for possible future restarts.
6. The manager sends all standard input data to the server and the server writes the standard input data to the Universal Spool database.
7. Once all standard input is spooled, the server starts the user process.
8. As the user process writes standard output and standard error data, the server writes the data to the Universal Spool database. If the manager is connected to the server, the data is written to the manager as well.
9. The user process executes until completion. Once the user process completes, the server writes the exit status of the user process to the Universal Command Server Database.
10. The server sends the exit status to the manager. This completes the unit of work.

## Case Example 2: Restart when User Process is Executing

The following figure diagrams the sequence of events that occur when a manager requests a restart of a currently executing unit of work. In this case the initial instance of the manager terminated. A restarted instance of the manager is started and requests to be reconnected to the unit of work.

This example continues from the first case example. Please refer to first case example for details of the component descriptions included in the diagram.

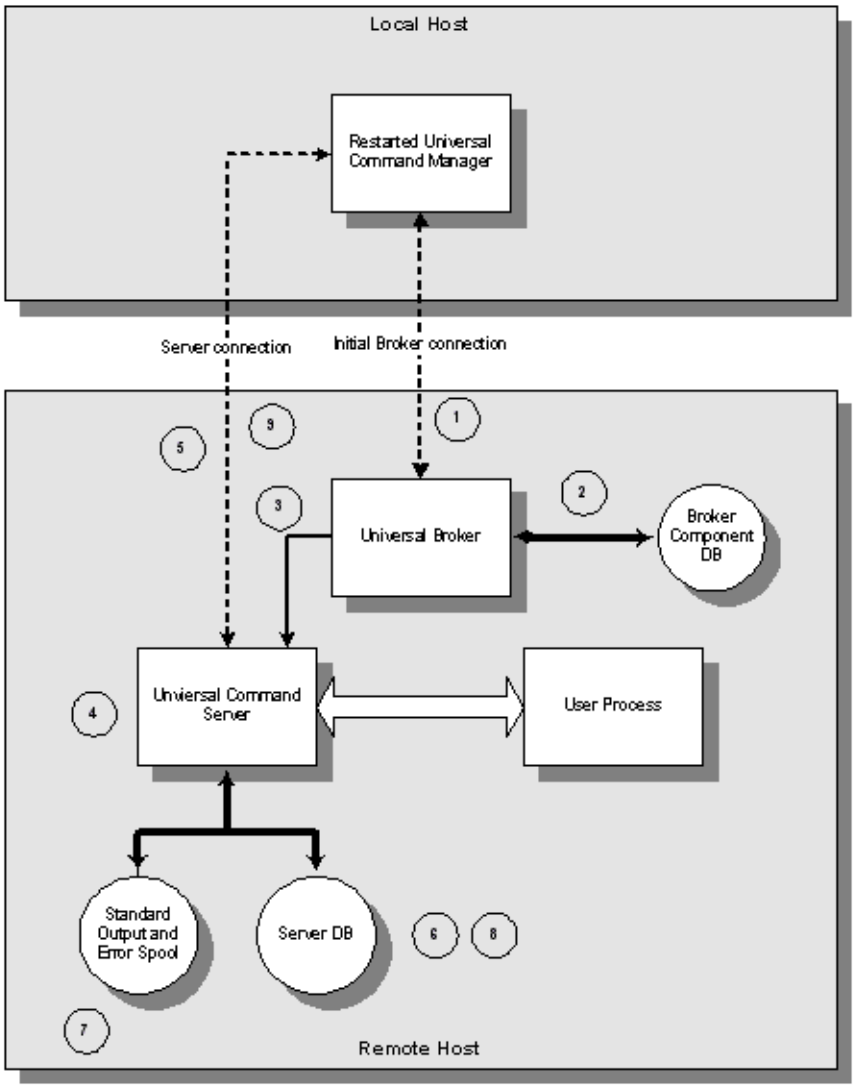


Figure 2.8 Case Example 2

The diagram demonstrates the sequence of events that occur when a manager requests to be restarted with a unit of work identified by a command ID. The numbers enclosed in circles represent the sequence of events and correspond to the listed descriptions below.

1. The restarted instance of the manager sends a restart request to the broker. The restart request contains the command ID specified as part of the invocation of the manager.
2. The broker verifies that the component is restartable and that the components communication state is acceptable for a restart request. If the server component were currently connected to a manager, its communication state would not permit a restart request.
3. The broker sends the restart request to the server corresponding to the command ID.
4. The server authenticates the request with the manager-supplied user ID and password. The password must be the same as the initial manager instance.
5. The manager and server exchange options that are used to carry out the request.
6. The server records the restart in the Universal Command Server Database.
7. The server sends spooled standard output and error files to the manager. This is performed while the user process may still be writing standard output and error to the spool. Once all spooled output is sent to the manager, the server will send standard output and error from the user process as it is being produced.
8. The user process executes to completion. The server records the user process exit status in the Universal Command Server Database.
9. The server sends the exit status to the manager. This completes the unit of work.

## Case Example 3: Restart when User Process has Ended

The following figure diagrams the sequence of events that occur when a manager requests a restart of a unit of work that has completed. In this case the initial instance of the manager has terminated, the user process completed normally, and a restarted instance of the manager is started and requests to be reconnected to the completed unit of work.

This example continues from the first case example. Please refer to first case example for details of the component descriptions included in the diagram.

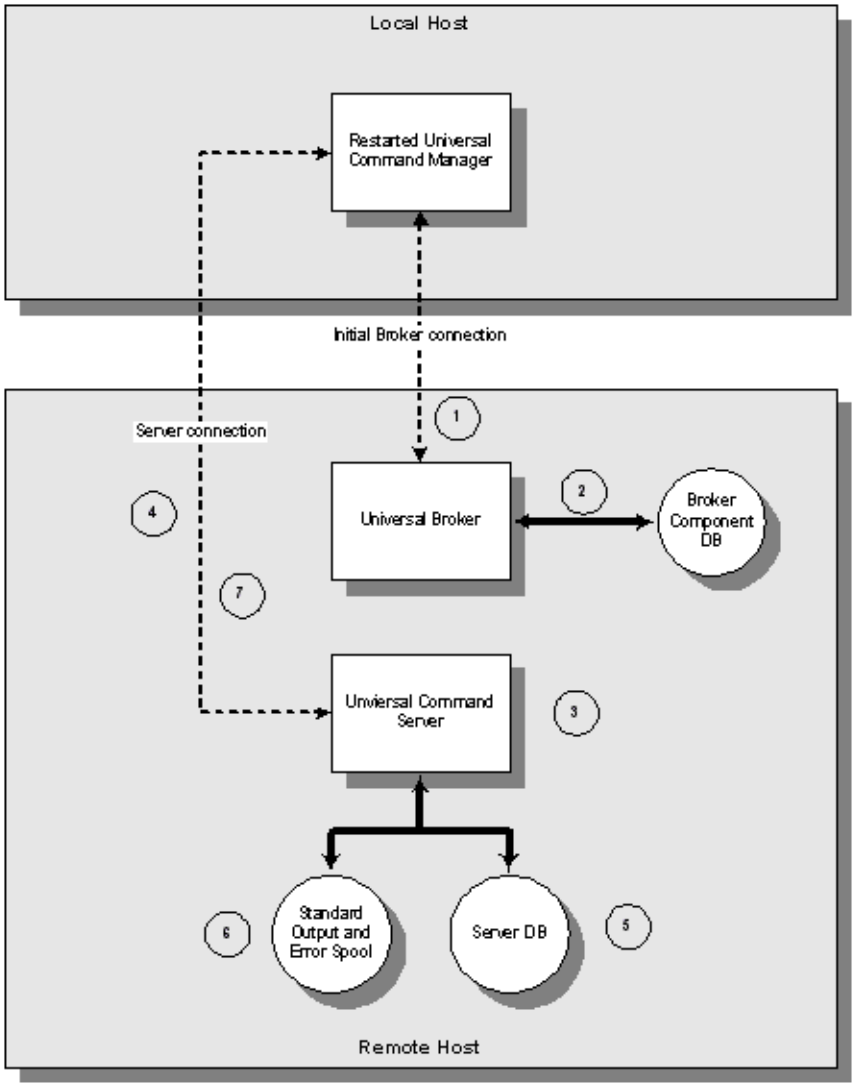


Figure 2.9 Case Example 3

The diagram demonstrates the sequence of events that occur when a manager requests to be restarted with a unit of work identified by a command ID. The user process in this case has completed execution. The numbers enclosed in circles represent the sequence of events and correspond to the following descriptions:

1. The restarted instance of the manager sends a restart request to the broker. The restart request contains the command ID specified as part of the invocation of the manager.
2. The broker verifies that the component is restartable and that the components communication state is acceptable for a restart request. If the server component were currently connected to a manager, its communication state would not permit a restart request.
3. Since the user process has completed, the broker starts a new server to process the restart request. The server authenticates the request with the manager-supplied user ID and password. The password must be the same as the initial manager instance.
4. The manager and server exchange options that are used to carry out the request.
5. The server records the restart in the Universal Command Server Database.
6. The server sends spooled standard output and error files to the manager.
7. The server sends the user process exit status to the manager. This completes the unit of work.

## 2.6.3 Component Management

In order to fully understand Universal Command fault tolerant features, some understanding of how the Universal Broker manages components is necessary.

Universal Broker manages component startup, execution, and termination. The broker and its components have the ability to communicate service requests and status information between each other.

The Broker maintains a database of components that are active or have completed and waiting for restart or reconnection. The component information maintained by the broker determines the current state of the component. This state information is required by the broker to determine if a restart or reconnect request from a manager is acceptable or not. The broker's component information can be viewed with the Universal Query program.

One piece of component information maintained by the broker is the component's communication state. The communication state primarily determines what state the Universal Command Server is in regarding its network connection with a manager and the completion of the user process and its associated spooled data.

Table 2.3, below, describes the communication state values.

- **Reconnect** column indicates whether or not a network reconnect request is valid.
- **Restart** column indicates whether or not a restart request is valid.

| State | Reconnect | Restart | Description |
|---|---|---|---|
| STARTED | NO | NO | The server has started.<br>If the server is restartable it is receiving the standard input file from the manager and spooling it. |
| ESTABLISHED | NO | NO | The server and manager are connected and processing normally. This state is the most common state when all is well. |
| DISCONNECTED | YES | YES | The server is not connected to the manager. This occurs when a network error has occurred, the manager halted, or the manager host halted.<br>The server is executing with either the network fault tolerant protocol, is restartable, or both.<br>Note: The server cannot tell if the manager is still executing or not since it cannot communicate with it. |
| ORPHANED | NO | YES | The manager has terminated. The manager sends a termination message to the server to notify it of its termination prior to terminating.<br>This state only occurs if the server is restartable. |
| RECONNECTING | NO | NO | The server has received a reconnect request from the manager to recover a lost network connection.<br>This state should not remain long, only for the time it takes to re-establish the network connections. |

| State | Reconnect | Restart | Description |
|---|---|---|---|
| RESTARTING | NO | NO | The server has received a restart request from the manager.<br>This state should not remain long, only for the time it takes to re-establish network connections. |
| PENDING | NO | YES | A restartable server and its user process have completed. The user process standard output and error files are in the spool.<br>A manager has not been restarted to pick up the spooled files and user process exit status. The server remains in this state until a manager is restarted. |
| COMPLETED | NO | NO | The server and manager have completed. All standard output and standard error files have been sent to the manager and the user process's exit status. |

Table 2.3 Component Communication States

# 2.7 z/OS CANCEL Command Support

The Universal Products provides network fault tolerance and, in some cases, manager fault tolerance (see Section 2.6 Fault Tolerance). These features provide users with the ability to execute jobs that will continue to run when the network is down and when a manager is terminated.

However, there are scenarios in which the user may want to cancel an executing job that supports manager and/or network fault tolerance and have both the manager and server processes terminate immediately. Because of fault tolerance, when the manager is terminated, the server side would begin a connection reestablishment protocol and continue to execute. This would allow the started user job to continue running.

In particular, z/OS supports a CANCEL command that will terminate a job executing on the z/OS operating system. When a Universal Command job is cancelled via the z/OS CANCEL command, the job terminates with either of these exit codes:

- Exit code S122, if it is cancelled with a dump.
- Exit code S222, if it is cancelled without a dump.

Part of the responsibility of a Universal Broker executing on a particular host is to monitor the status of all locally running manager processes on that machine. So, when instructed, that Universal Broker could issue a STOP command to the Universal Command Server process associated with the stopped/ended manager process.

## 2.7.1 Exit Codes

Through the use of the SERVER_STOP_CONDITIONS configuration option, the Universal Command Manager process notifies the locally running Universal Broker of the exit codes that should cause it to terminate the running Server process. With this option, the user can specify a list of exit codes that should trigger the locally running Universal Broker to issue the STOP command to the manager's Universal Command server-side process.

SERVER_STOP_CONDITIONS can specify a single exit code or a comma-separated list of exit codes. These stop conditions are passed from the manager to the locally running Universal Broker, which store this and other component-specific data about the executing manager component. When this executing Universal Command Manager process is cancelled or stopped, the locally running Universal Broker detects the ending of the manager process and retrieves its process completion information, which includes the exit code of the manager.

The Universal Broker then compares this exit code with the list of exit codes provided by SERVER_STOP_CONDITIONS. If a match is found, and either network fault tolerance or manager fault tolerance is enabled, the Universal Broker will execute a **`uctl`** command to STOP the running Universal Command Server component.

## 2.7.2 Security Token

For security purposes, Universal Products pass around a security token that is used by the locally running Universal Broker to STOP associated Universal Command Server process.

This security token is generated on a component-by-component basis by the Universal Broker process that starts the Universal Command Server. Upon generation, this token is returned to the Universal Command Manager which, in turn, updates its locally running Universal Broker with this token. The locally running Universal Broker then uses this token with the issued STOP command to cancel the running Universal Command Server process.

When this token is received by the Universal Broker processes with the request to STOP the server component, the Broker authenticates the received token with the stored token for the running Universal Command Server process. When the token is authenticated, the Universal Command Server process is STOPPED.

# 2.8 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 UACL section for each operating system in this 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 of the Universal Products Utilities 3.2.0 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.

## 2.8.1 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 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 HP NonStop configuration files. See Section 2.2.6 Configuration File Syntax for details.

## 2.8.2 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:

```
ucmd_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 2.4, 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).<br>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 (`.`).<br>Many RDN values can be used in a DN. Some of the most common values are:<br>• C Country name<br>• CN Common name<br>• L Locality<br>• O Organization<br>• OU Organizational Unit<br>• ST State<br>The RDN attributes must be listed in the same order as they are defined in the certificate to be considered matched.<br>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.<br>The RDN attribute values can include pattern matching characters. An asterisk (`*`) matches 0 or more characters and a question mark (`?`) matches one character.<br>Some example of SUBJECT values are:<br>• **`subject="C=US,ST=Georgia,O=Acme,CN=Road Runner"`**<br>• **`subject="C=US,ST=Georgia,O=Acme,CN=Road * "`**<br>• **`subject="C=US,ST=Georgia,O=Acme,CN=Road ?unner"`**<br>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**.<br>The attribute values may include pattern matching characters. An asterisk (*) matches 0 or more characters and a question mark (?) matches one character.<br>Some example EMAIL values are:<br>• **email=user1@acme.com**<br>• **email=*@acme.com**<br>• **email=user?@acme.com**<br>RFC 822 names are not case sensitive. |
| HOSTNAME | Matches the following X.509 fields in the order listed:<br>1. **dNSName** of the **subjectAltName** extension value.<br>2. **commonName** (CN) RDN attribute of the **subject** field's DN value.<br>Some example HOSTNAME values are:<br>• **hostname=bigfish.acme.com**<br>• **hostname=*.acme.com**<br>The values are not case sensitive. |
| IP ADDRESS | Matches the X.509 **iPAddress** field of the **subjectAltName** extension value.<br>An example IPADDRESS value is:<br>• **ipaddress=10.20.30.40** |
| SERIAL NUMBER | Matches the X.509 **serialNumber** value.<br>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.<br>An example SERIALNUMBER value is:<br>• **serialnumber=0x7a2d52cbae** |

Table 2.4 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 2.5, 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:<br>• 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**.<br>• The comparison is case insensitive.<br>• 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 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 2.6, 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.<br>The HOST value has the following syntax:<br>• Dotted numeric form of an IP address. For example, **10.20.30.40**.<br>• 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.<br>• 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.<br>Note: Contact your network administrator for calculation of the correct net / mask expression.<br>• Host name for an IP address. For example, **sysa.abc.com**.<br>• 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.<br>• 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.<br>The REMOTE_USER value has the following syntax:<br>• 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**.<br>• 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**.<br>• 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 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 2.7, 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:<br>• 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**.<br>• 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**.<br>• 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**.<br>• 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.<br>A space character delimits the variable name, or it can be enclosed in parentheses (for example, **$(RMTUSER)**), in which case it is delimited by the right parenthesis. This is useful if it is immediately followed by text.<br>For example, if the remote user name is **TOM**, a LOCAL_USER value of **$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**.<br>**Windows**<br>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:<br>• 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**.<br>• The comparison is case insensitive.<br>• 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:<br>• 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**.<br>• 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.<br>• 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**.<br>• 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**.<br>• 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 2.7 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.

# 2.9 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 Products documentation for additional details.)

## 2.9.1 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 2.9.3 Message Levels).

## 2.9.2 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.

## 2.9.3 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.

# 2.9.4 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 message destination values depend on the host operating system.

## z/OS Message Destinations

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 following table.

| Destination | Description |
|---|---|
| LOGFILE | Messages are written to ddname UNVLOG.<br>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

Message destinations are listed in the following table.

| 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.<br>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.<br>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

Message destinations are listed in the following table.

| 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.<br>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

Message destinations are listed in the following table.

| 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

Message destinations are listed in the following table.

| Destination | Description |
|---|---|
| STDERR | Messages are written to standard error. |
| LOGFILE | Messages are written to a log file. Not all programs provide this destination.<br>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. |

# 2.10 Command References

Command references are predefined commands or scripts saved in files on a Universal Command Server system.

Universal Command Managers can request command references to be executed by specifying a COMMAND_TYPE value of **`cmdref`**. The Manager specifies the command reference by name. The Server finds the command reference file and executes the command or script contained within the command reference.

A command reference provides the ability to precisely define and control what is executed by the Server. The Manager does not provide any commands or scripts; everything is defined within the command reference. The command reference may optionally be defined to accept command or script options from the Manager.

## 2.10.1 Command Reference Syntax

A command reference file is a sequential file containing two sections:

1. Options Section
2. Command Section

### Options Section

The Options section contains Command Reference Options that define the command reference and specify operational characteristics. Comments also can be included in the Options section.

The Options section syntax rules are:

- Lines starting with a hash character (#) are comments.
- Blank lines are ignored.
- Options are specified with a Name followed by a Value.
- Names are prefixed with a hyphen (-).
- Names are case-insensitive.
- Values are case-insensitive.
- One or more spaces must separate the Name and Value.
- Options section is ended with <eof> at the start of a line.

### Command Reference Options

Table 2.8, below, identifies all valid command reference options.

| Name | Description | Values |
|---|---|---|
| -format | Format of the command section.<br>This option is required. | Valid values are:<br>• **cmd** Single command is specified.<br>• **script** Script is specified. The script is defined in the same format as if it was being defined normally in a self-contained file. |
| -type | Type of command or script. | Possible values depend on the Universal Command Server operating system.<br>For example, on Windows, possible values for scripts are:<br>• **shell** for commands<br>• file extension with a program association for scripts.<br>See the Universal Command Server COMMAND_TYPE and SCRIPT_TYPE options for description of valid values. |
| -allow_options | Specification for whether or not options provided by the Manager are passed to the command or script. | Valid values are:<br>• **yes** Options are passed.<br>  • For commands, the Manager options are concatenated to the end of the command.<br>  • For scripts, the Manager options are passed to the script as command line arguments.<br>• **no** Options are not passed.<br>Default is **no**. |

Table 2.8 Command Reference Options

## Command Section

The Command section provides the actual command or script to be executed. Whether it is interpreted as a command or script is specified by the **`-format`** option.

### Commands

Commands are single commands that are executed based on the **`-type`** option.

The command may span multiple lines. No line continuation characters are required. All lines will be concatenated together to form one command.

### Scripts

Scripts are read verbatim and placed into a temporary file. The temporary file is passed to the script processor for the appropriate script type, as specified by the **`-type`** option.

### Command Reference Example

Figure 2.10, below, is an example command reference that executes a Windows DIR command.

Note: The two sections are separated with a logical end-of-file marker, <eof>.

```
# Comments describing the command reference.
#
# Execute the DIR command.
#

-format         cmd
-type           shell
-allow_options  no

<eof>

DIR
```

Figure 2.10 Command Reference Syntax

## 2.10.2 Configuration

Command reference files are located in a location defined by the Universal Command Server CMD_REFERENCE_DIRECTORY option. When a Manager requests a command reference execution, the Server searches for the command reference in the configured location and executes it.

Command references can be protected with Universal Access Control List (UACL) rules. Access can be denied or allowed based on the command reference name.

See the Security section of the appropriate Server platform for details.

## 2.10.3 Command Reference Request

A Universal Command Manager requests the execution of a command reference just as it does for other command types: via the COMMAND_TYPE option.

Note: Universal Command Managers earlier than version 3.1.0 must use the SERVER_OPTIONS option to specify a command type.

### Examples

The following examples illustrate how a Universal Command Manager for UNIX would invoke a command reference.

3.1.0 and later: no options

```
ucmd –cmd cmd100 –cmd_type cmdref ...
```

3.1.0 and later: with options

```
ucmd –cmd “cmd100 opt1 opt2” –cmd_type cmdref ...
```

Earlier than 3.1.0: no options

```
ucmd –cmd cmd100 -server " –cmd_type cmdref" ...
```

Earlier than 3.1.0: with options

```
ucmd –cmd “cmd100 opt1 opt2” -server " –cmd_type cmdref" ...
```

The actual format of the options, **opt1** and **opt2**, depend on the command or script being executed in the command reference. In some cases, the options may be required to be comma separated and, other cases, space separated.

Note: For Universal Command Managers earlier than 3.1.0, the **-server** value must include at least one space between the first quotation mark and the following option (in these examples, **-cmd_type**).

# 2.11 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.

## 2.11.1 Sample Certificate Directory

Figure 2.11, below, provides a sample diagram of a small X.500 directory.

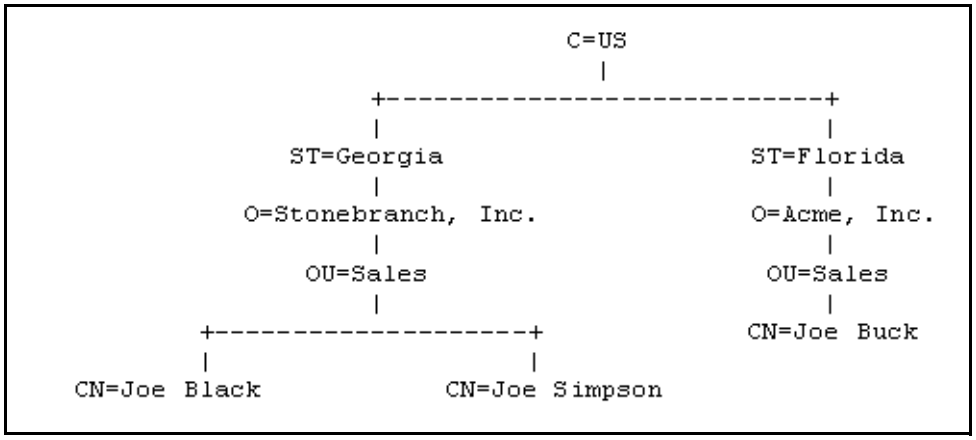


Figure 2.11 X.500 Directory (sample)

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.

## 2.11.2 Sample X.509 Certificate

Figure 2.12, 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 2.12 X.509 Version 3 Certificate (sample)

Note: The contents of a certificate file does not look like the information in Figure 2.12, which is produced by a certificate utility that uses the certificate file as input. Certificates can be saved in multiple file formats, so their file contents will look very different.

## 2.11.3 Certificate Fields

A certificate is composed of many fields.

Table 2.9, below, describes the main certificate fields.

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

Table 2.9 Certificate Fields

## 2.11.4 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. So 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 needs to 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 fully 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 website.

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 Chapter 3 Universal Certificate in the Universal Products Utilities 3.2.0 User Guide for details.

# Chapter 3
# Universal Command Manager for z/OS

## 3.1 Overview

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

UCMD Manager executes commands on any computer running the UCMD Server component.

You indicate to the UCMD Manager what command(s) to execute and how the standard input and output and error data should be processed. The UCMD Manager connects to the UCMD Server and processes your request.

The z/OS Batch Manager provides a batch job interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered them directly on the remote command line. Standard input and output and error files are supplied using JCL DD statements.

UCMD Manager registers with a locally running Universal Broker. Consequentially, a Universal Broker must be running in order for a UCMD Manager to execute.

# 3.2 Usage

UCMD Manager for z/OS executes as a batch job. It consists of:

- Batch JCL
- Configuration options

This section describes the JCL, configuration and configuration options, and command line syntax of UCMD Manager for z/OS.

Section 3.4 Examples of UCMD Manager for z/OS provides examples demonstrating the flexibility of Universal Command.

## 3.2.1 JCL Procedure

Figure 3.1, below, illustrates the UCMD Manager for z/OS JCL procedure (**UCMDPRC**, located in the **SUNVSAMP** library) that is provided to simplify the execution JCL and future maintenance.

```
//UCMDPRC  PROC UPARM=,                -- UCMD options
//             STDOUT='SYSOUT=*',      -- stdout of remote command
//             STDERR='SYSOUT=*',      -- stderr of remote command
//             STDIN='DUMMY',          -- stdin of remote command
//             UCMDPRE=#SHLQ.UNV
//*
//PS1      EXEC PGM=UCMD,PARM='ENVAR(TZ=EST5EDT)/&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UCMDPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UCMDPRE..SUNVNLS
//UNVTRACE DD  SYSOUT=*
//*
//UNVOUT   DD  &STDOUT                          -- Remote stdout
//UNVERR   DD  &STDERR                          -- Remote stderr
//UNVIN    DD  &STDIN                           -- Remote stdin
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
```

Figure 3.1 Universal Command Manager for z/OS – JCL Procedure

The procedure provides the parameters **STDIN**, **STDOUT**, and **STDERR** to specify the **stdin**, **stdout**, and **stderr** files, respectively, of the remote command. The **UPARM** parameter is used to specify EXEC PARM keyword values for the UCMD program. 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.

The **UCMDPRE** parameter specifies the data set name prefix of Universal Products installation data sets.

## 3.2.2 DD Statements used in JCL Procedure

Table 3.1, below, describes the DD statements used in the UCMD Manager for z/OS JCL procedure illustrated in Figure 3.1.

| ddname | DCB Attributes * | Mode | Description |
|---|---|---|---|
| STEPLIB | DSORG=PO,<br>RECFM=U | input | Universal Products load library containing the program being executed. |
| UNVNLS | DSORG=PO,<br>RECFM=(F, FB, V, VB) | input | Universal Products national language support library. Contains message catalogs and code page translation tables. |
| UNVTRACE | DSORG=PS,<br>RECFM=(F, FB, V, VB) | output | UCMD trace output. |
| UNVOUT | DSORG=PS,<br>RECFM=(F, FB, V, VB) | output | Remote command's stdout file. When the remote command writes to its stdout file, this is the file to which it writes. |
| UNVERR | DSORG=PS,<br>RECFM=(F, FB, V, VB) | output | Remote command's stderr file. When the remote command writes to its stderr file, this is the file to which it writes. |
| UNVIN | DSORG=PS,<br>RECFM=(F, FB, V, VB) | input | Remote command's stdin file. When the remote command reads from its stdin file, this is the file from which it reads. |
| SYSPRINT | DSORG=PS,<br>RECFM=(F, FB, V, VB) | output | Standard output file for the UCMD program. UCMD does not write any messages to SYSPRINT. |
| SYSOUT | DSORG=PS,<br>RECFM=(F, FB, V, VB) | output | Standard error file for the UCMD program. UCMD writes its messages to SYSOUT. |
| * The C runtime library determines the default DCB attributes. See the IBM manual *OS/390 C/C++ Programming Guide* for details on default DCB attributes for stream I/O. | | | |

Table 3.1 Universal Command for z/OS – DD Statements in JCL Procedure

### DD Statement Categories

UCMD Manager DD statements are organized into the following categories:

- Runtime specifications
  STEPLIB and UNVNLS are in this category.
- Remote command standard files (stdin, stdout and stderr)
  UNVIN, UNVOUT and UNVERR are in this category.
- UCMD message and command files (stdin, stdout and stderr)
  SYSPRINT, SYSOUT, and UNVTRACE are in this category.
- Command files
  User-defined DD statements to which Universal Command commands refer.

## 3.2.3 JCL

Figure 3.2, below, illustrates the UCMD Manager for z/OS JCL using the **UCMDPRC** procedure illustrated in Figure 3.1.

```
//jobname  JOB CLASS=A,MSGCLASS=X
//STEP1    EXEC UCMDPRC,
//         STDOUT='DISP=SHR,DSN=my.local.file'
//SYSIN    DD  *
 -copy 'ucopy file' -host dallas -userid joe -pwd akkSdiq
/*
```

Figure 3.2 Universal Command Manager for z/OS – JCL

Job step STEP1 executes the procedure **UCMDPRC** with the **STDOUT** parameter set to a value that allocates a local file.

The command options are specified on the SYSIN DD.

## 3.2.4 Configuration

Configuration consists of:

- Setting default options and preferences for all executions of UCMD Manager.
- Setting options and preferences for a single execution of UCMD 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; PARM keyword options 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 UCMD Manager configuration file is provided to the manager by the local Universal Broker with which it registers. The UCMD Manager configuration file is located in the **UCMCFG00** member of the PDSE allocated to the **UNVCONF** ddname in the Universal Broker started task.

The configuration file 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 complete details on configuration methods and command input for Universal Products.

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

# 3.2.5 Configuration Options

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

## Configuration Options Categories

Table 3.2, below, categorizes the configuration options into logical areas of application.

| Category | Description |
|---|---|
| Certificates | X.509 certificate-related options. |
| Command | Command or script to execute on the remote system. If a script is being executed, the script may reside on the local host on which the Manager is running or the remote host on which the Server is running.<br>It also includes options to control the process environment in which the command executes. |
| Events | Options used to define event generation. |
| Local | Options required for local Broker registration. |
| Messages | Universal Command message options. |
| Miscellaneous | Options used to display command help and program versions. |
| Network | Processing options for all the data transferred between the remote and local systems. |
| Options | Alternative methods to specify configuration options. |
| Remote | Network address of the remote system and connection options. |
| Spool | Options that control whether or not the Manager spools its standard files and how they are processed. |
| Standard File | Processing options for the standard files transferred between the remote and local systems.<br>The STDFILE options are specified differently then the other options. There are three types of standard files: stdin, stdout, and stderr. Each standard file can have a different set of options applied. In order to distinguish between the standard files, the options must start with a standard file specification option (STDERR_SPEC, STDIN_SPEC, or STDOUT_SPEC). The standard file options (names starting with **SIO**_) follow the standard file specification option. Section 3.4 Examples of UCMD Manager for z/OS demonstrates this. |
| User | User account that the command executes with on the remote system. |

Table 3.2 Universal Command Manager for z/OS - Configuration Options Categories

The UCMD 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 Command 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 | ddname of the PEM-formatted CRL.. |
| PRIVATE_KEY | ddname of Manager's PEM formatted RSA private key. |
| PRIVATE_KEY_PWD | Password for the Manger's PRIVATE_KEY. |
| SAF_KEY_RING | SAF certificate key ring name. |
| SAF_KEY_RING_LABEL | SAF key ring certificate label. |
| SSL_IMPLEMENTATION | SSL implementation. |
| VERIFY_HOST_NAME | Specifies that the Broker's X.509 certificate host name field must be verified. |
| VERIFY_SERIAL_NUMBER | Specifies that the Broker's X.509 certificate serial number field must be verified. |

## Command Category Options

| Option Name | Description |
|---|---|
| COMMAND | Remote command to execute. |
| COMMAND_ID | Unique command ID associated the unit of work. |
| COMMAND_TYPE | Type of command specified with option COMMAND. |
| EXIT_CODE_MAP | Allows exit codes from the user process executed by UCMD Server to be translated (mapped) to a corresponding exit code for UCMD Manager. |
| LOGIN | Specifies whether or not the command runs in a login environment. |
| MANAGER_FAULT_TOLERANT | Specification for whether or not the manager fault tolerant feature is used. |
| SCRIPT_FILE | Local script file to execute on the remote system. |
| SCRIPT_OPTIONS | Command line options passed to the script file. |
| SCRIPT_TYPE | Type of script file specified by option SCRIPT_FILE. |

## 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 UCMD Manager must register. |

## Messages Category Options

| Option Name | Description |
|---|---|
| MESSAGE_LANGUAGE | Language of messages written. |
| MESSAGE_LEVEL | Level of messages written. |
| TRACE_FILE_LINES | Maximum number of lines written to a trace file before it wraps around. |
| TRACE_TABLE | Memory trace table specification. |

## Miscellaneous Category Options

| Option Name | Description |
|---|---|
| COMMENT | User-defined string. |
| HELP | Write command option help. |
| VERSION | Write program version. |

## Network Category Options

| Option Name | Description |
|---|---|
| CODE_PAGE | Code page used for text translation. |
| CTL_SSL_CIPHER_LIST | SSL cipher list for the control session. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the data sessions. |
| DEFAULT_CIPHER | Default SSL cipher used for data sessions. |
| FORCE_COMPLETE | Forces a manager fault tolerant server in a PENDING communication state to COMPLETED state without retrieving the spooled data. |
| JOB_RETENTION | Specifies how long a restartable Server waits for a reconnect after the user process completes. |
| NETWORK_DELAY | Maximum number of seconds considered acceptable to wait for data communications. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not the network fault tolerant protocol is used. |
| RECONNECT_RETRY_COUNT | Maximum number of network fault tolerant reconnect attempts. |
| RECONNECT_RETRY_INTERVAL | Number of seconds between network fault tolerant reconnect attempts. |
| RESTART | Specification for whether or not the manager is requesting restart. |
| SERVER_STOP_CONDITIONS | Exit codes that cause Universal Broker to cancel the corresponding UCMD Server of the exited UCMD Manager. |

## Options Category Options

| Option Name | Description |
|---|---|
| ASSIGN_PROCESS_TO_JOB | Specification for whether or not UCMD Server assigns child processes to a single Windows job object. |
| 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. |
| SERVER_OPTIONS | Universal Command Server options that can be overridden by Managers. |

## Remote Category Options

| Option Name | Description |
|---|---|
| CONNECT_TIMEOUT | Amount of time that a UCMD Manager will wait for a connection to a remote Universal Broker to complete. |
| DNS_EXPAND | Number of IP addresses returned to UCMD Manager following a DNS query issued to resolve a host name. |
| HOST_SELECTION | Host in the REMOTE_HOST list that the UCMD Manager will choose to begin its attempts to connect to a remote Universal Broker. |
| HOSTNAME_RETRY_COUNT | Number of times that UCMD will attempt to resolve the host name of a specified Universal Broker before it ends with a connect error. |
| MFT_SAFE_MODE | Situations in which more than one host may be specified in the REMOTE_HOST list when manager fault tolerance (MFT) is enabled. |
| OUTBOUND_IP | Host or IP address to use for all outgoing IP connections. |
| REMOTE_HOST | List of one or more hosts upon which a command may run. |
| REMOTE_PORT | TCP/IP port number of the remote Broker. |

## Standard File Category Options

| Option Name | Description |
|---|---|
| SIO_DATA_AUTHENTICATION | Specifies if data integrity checks are performed on a standard file. |
| SIO_DATA_COMPRESSION | Specifies if and how data is compressed on a standard file. |
| SIO_DATA_ENCRYPTION | Specifies if data is encrypted on a standard file. |
| SIO_LOCAL_CODE_PAGE | Code page used for local text translation on a standard file. |
| SIO_LOCAL_FILE | Local file used for a standard file instead of the default. |
| SIO_MODE | Translation mode of a standard file. |
| SIO_REMOTE_CODE_PAGE | Code page used for remote text translation on a standard file. |
| SIO_TRAILING_SPACES | Read trailing spaces of z/OS fixed format records. |
| STDERR_FILE_SPEC | Start of standard error file specification options. |
| STDIN_FILE_SPEC | Start of standard input file specification options. |
| STDOUT_FILE_SPEC | Start of standard output file specification options. |

## User Category Options

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

## 3.2.6 Command Line Syntax

Figure 3.3 and Figure 3.4, below, illustrate the command line syntax — using the command line, long form of the configuration options — of UCMD Manager for z/OS.

```
ucmd
{ -cmd command [-cmd_type {cmdref|shell|stc} ] | -script ddname
   [-options options] [-script_type type] }
-host hostlist
[-connect_timeout seconds]
[-dns_expand {yes|no} ]
[-host_selection {sequential|random} ]
[-mft_safe_mode {yes|no} ]
[-file ddname | -encryptedfile ddname [-key key] ]  *
[-port port]
[-system_id ID]
[-userid user [-pwd pwd] ]
[-hostname_retry_count count]
[-outboundip host]
[-server options]
[-assign_process_to_job option]
[-managerft {yes|no} ]
[-cmdid id]
[-login {yes|no} ]
[-lang language]
[-level {trace|audit|info|warn|error}[,{time|notime} ]
[-tracefilelines lines]
[-trace_table size,{error|always|never} ]
[-ssl_implementation {openssl|system} ]
[-ca_certs ddname [-verify_host_name {yes|no|hostname} ]
   [-verify_serial_number number] ]
[-cert ddname -private_key ddname [-private_key_pwd password] ]
[-crl ddname]
[-saf_key_ring name [-saf_key_ring_label label] ]
[-ctl_ssl_cipher_list cipherlist]
[-data_ssl_cipher_list cipherlist]
[-default_cipher cipher]
[-forcecomplete {yes|no} ]
[-job_retention seconds]
[-delay seconds]
[-networkft {yes|no} ]
[-retry_count number]
[-retry_interval seconds]
```

Figure 3.3 Universal Command for z/OS - Command Line Syntax (1 of 2)

```
[-restart {yes|no|auto} [-managerft {yes|no} [-cmdid id] ] ]
[-server_stop_conditions codes]
[-codepage codepage]
[-compress {yes|no}[,{zlib|hasp} ] ]
[-encrypt {yes|no} ]
[-authenticate {yes|no} ]
[-stdin | -stdout | -stderr]
   [-codepage codepage]
   [-compress {yes|no}[,{zlib|hasp} ] ]
   [-encrypt {yes|no} ]
   [-authenticate {yes|no} ]
   [-localfile ddname]
   [-mode {text|binary}[,{ucs|direct} ] [-trailingspaces {yes|no} ] ]
   [-remotecodepage codepage]
[-exit_code_map map]
[-comment text]

ucmd
{-help | -version}
```

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

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

# 3.3 Shutdown Procedure

When the UCMD Manager and UCMD Server are configured to use the network fault tolerant protocol, the z/OS Manager must be shut down properly in order for the UCMD Server and user command to shut down as well. If not properly shut down, the UCMD Server will wait for a period of time for the UCMD Manager to reestablish network connections.

If the UCMD Manager and UCMD Server are not configured to use the network fault tolerant protocol, the z/OS Manager can be shut down with any available method.

## 3.3.1 Fault Tolerant Shutdown

The z/OS Manager is a z/OS UNIX System Services (USS) application. A shutdown sequence is initiated by sending a USS termination signal to the job. The UCMD Manager detects the signal and promptly sends a terminate message to the UCMD Server. Upon receiving the terminate message, the UCMD Server starts its shutdown sequence.

A termination signal is sent to a z/OS Manager, as shown in the following steps:

| Step | Instruction |
|---|---|
| Step 1 | For the z/OS Manager job to be shut down, obtain its USS process identifier (PID). The PID of a job is displayed with the DISPLAY OMVS console command:<br>`D OMVS,U=`*`userid`*<br>( *`userid`* is the user identifier with which the batch job executes.)<br>If you do not know the user identifier, all USS work can be displayed with the following command:<br>`D OMVS,A=ALL`<br>The batch job's name and address space identifier are listed. Find the job name and address space identifier of the job to be terminated and obtain its process identifier from the PID column. |
| Step 2 | Send a termination signal to the process identifier with the MODIFY BPXOINIT command:<br>`F BPXOINIT,TERM=`*`pid`*<br>(*`pid`* is the process identifier obtained from the previous step.) |

## 3.3.2 Non-Fault Tolerant Shutdown

Since the UCMD Server shutdown sequence is started as soon as a network connection is prematurely closed, the z/OS Manager shutdown can be initiated with any available z/OS job termination method.

The most common and safest method is the CANCEL console command:

```
C jobname
```

(**jobname** is the name of the z/OS Manager batch job to terminate.)

# 3.4 Examples of UCMD Manager for z/OS

Appendix A Examples provides operating system-specific examples that demonstrate the use of Universal Command.

Included in this appendix are the following examples that demonstrate the use of Universal Command Manager for z/OS:

- Directory Listing for UNIX Server
- Directory Listing for Windows Server
- Netstat Command for Windows
- Netstat Command for UNIX
- z/OS to UNIX Backup to z/OS Dataset
- z/OS to UNIX Backup Restore from z/OS Dataset
- UNIX Copy to z/OS Dataset
- Script Commands to Windows
- JCL Procedure
- Manager Command to Windows, UNIX, OS/400, HP NonStop
- Command Coded as a Script
- Override Standard Files with DDNAME
- Override Standard Files with SYMBOLICS
- Override Command Line Parameters from the Execute Statement
- Authentication Parameters from Encrypted File
- Override Restart Parameter in Command Line
- Override Restart Parameter in PARM
- Redirect Standard Out to File and UCMD Manager
- Execute a Windows .bat file
- Unique Command ID with CA-7 or CA-Scheduler
- Unique Command ID with Zeke
- Unique Command ID with OPC

# 3.5 Security

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

UCMD security concerns are:

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

## 3.5.1 Data Set Permissions

Only trusted user accounts should have write access to the UUCMD Manager installation files. Eligible users of UUCMD require read access to the national language support library SUNVNLS, the configuration file UNVCONF, and the load library SUNVLOAD.

## 3.5.2 RACF Protection

The UCMD Manager verifies a users access to a RACF general resource profile. The resource profile controls a user's access to execute a command on a remote host with a specific remote user identity.

See the Universal Products 3.2.0 Installation Guide for complete details on installing and administering UCMD Manager RACF profiles.

# Chapter 4
# Universal Command Manager for Windows

## 4.1 Overview

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

UCMD Manager provides a command line interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered the command directly on the remote command line.

On the command line, you must specify a command to execute and a remote Universal Broker. Additional input to each execution of the UCMD Manager command is made via configuration options, which control product behavior and resource allocation for that execution.

Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

UCMD Manager registers with a locally running Universal Broker. Consequentially, a Universal Broker must be running in order for a UCMD Manager to execute.

# 4.2 Usage

UCMD Manager for Windows executes as a command line application. It consists of the command line program followed by a list of configuration options. This section describes the command input, configuration and configuration options, and command line syntax.

Section 4.3 Examples of UCMD Manager for Windows provides some examples that demonstrate the flexibility of Universal Command.

## 4.2.1 Standard Input

The UCMD Manager command is executed from the Command Prompt window or a batch file. The **ucmd** command reads from standard input and writes it to the UCMD Server for the remote command to read as its standard input.

When UCMD Manager is executed from the Command Prompt window, standard input is allocated to the window itself. Any characters typed in the Command Prompt window are read as standard input by **ucmd** and transmitted to the UCMD Server.

If **ucmd** is executing a remote command that is reading standard input, it will read the characters being typed in the Command Prompt window until it receives an end-of-file indicator. To enter end-of-file in a Command Prompt window, press <Ctrl+Z> <Enter> at the start of a new line.

The allocation of standard input can be changed with a Command Prompt redirection operator. The redirection operators instruct Windows to change the allocation of the standard files.

To change the allocation of standard input, use the **<** operator. Windows can redirect to a special file referred to as NUL. The NUL file is always empty if read from, and never full if written to (all data written to NUL is never saved on disk or in memory).

To allocate standard input to NUL, the command syntax is as follows:

```
ucmd [OPTIONS...] < NUL
```

## 4.2.2 Configuration

Configuration consists of:

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

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

# 4.2.3 Configuration Options

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

## Configuration Options Categories

Table 4.1, below, categorizes the configuration options into logical areas of application.

| Category | Description |
|---|---|
| Certificates | X.509 certificate-related options. |
| Command | Command or script to execute on the remote system. If a script is being executed, the script may reside on the local host on which the UCMD Manager is running or the remote host on which the UCMD Server is running.<br>The command category also includes options to control the process environment in which the command executes. |
| Events | Options used to define event generation. |
| Installation | Options that specify installation requirements, such as directory locations. |
| Messages | Universal Command message options. |
| Miscellaneous | Options use to display command help and program versions. |
| Network | Processing options for all the data transferred between the remote and local systems. |
| Options | Alternative methods to specify command options. |
| Remote | Network address of the remote system and connection options. |
| Spool | Options that control whether or not the UCMD Manager spools its standard files and how they are processed. |
| Standard File | Processing options for the standard files transferred between the remote and local systems.<br>The STDFILE options are specified differently then the other options. There are three types of standard files: stderr, stdin, and stdout. Each standard file can have a different set of options applied. In order to distinguish between the standard files, the options must start with a standard file specification option (STDIN_SPEC, STDOUT_SPEC, or STDERR_SPEC). The standard file options (names starting with SIO_) follow the standard file specification option. (Section 4.3 Examples of UCMD Manager for Windows demonstrates this.) |
| User | User account that the command executes with on the remote system. |

Table 4.1 Universal Command Manager for Windows - Command Options Categories

The UCMD Manager for Windows 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 Command 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 for whether or not the Broker's X.509 certificate host name field must be verified. |
| VERIFY_SERIAL_NUMBER | Specification for whether or not the Broker's X.509 certificate serial number field must be verified. |

## Command Category Options

| Option Name | Description |
|---|---|
| COMMAND | Remote command to execute. |
| COMMAND_ID | Unique command ID associated the unit of work. |
| COMMAND_TYPE | Type of command specified with option COMMAND. |
| EXIT_CODE_MAP | Allows exit codes from user process executed by UCMD Server to be mapped to a corresponding exit code for UCMD Manager |
| LOGIN | Specification for whether or not the command runs in a login environment. |
| MANAGER_FAULT_TOLERANT | Specification for whether or not manager fault tolerance is used. |
| SCRIPT_FILE | Local script file to execute on the remote system. |
| SCRIPT_OPTIONS | Command line options passed to the script file. |
| SCRIPT_TYPE | Type of script file specified by option SCRIPT_FILE. |

## 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 UCMD Manager 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 |
| TRACE_FILE_LINES | Maximum number of lines written to a trace file before it wraps around. |
| TRACE_TABLE | Memory trace table specification. |

## Miscellaneous Category Options

| Option Name | Description |
|---|---|
| COMMENT | User-defined string |
| HELP | Write command option help. |
| VERSION | Write program version. |

## Network Category Options

| Option Name | Description |
|---|---|
| CODE_PAGE | Code page used for text translation. |
| CTL_SSL_CIPHER_LIST | SSL cipher list for the control session. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the data sessions. |
| DEFAULT_CIPHER | Default SSL cipher used for data sessions. |
| FORCE_COMPLETE | Forces a manager fault tolerant server in a PENDING communication state to COMPLETED state without retrieving the spooled data. |
| JOB_RETENTION | Length of time that a Server waits for a reconnect after the user process completes. |
| NETWORK_DELAY | Maximum number of seconds considered acceptable to wait for data communications. |
| NETWORK_FAULT_TOLERANT | Specification for whether network fault tolerant protocol is used. |
| RECONNECT_RETRY_COUNT | Maximum number of network fault tolerant reconnect attempts. |
| RECONNECT_RETRY_INTERVAL | Number of seconds between network fault tolerant reconnect attempts. |
| RESTART | Specifies if the manager is requesting restart or not. |

## Options Category Options

| Option Name | Description |
|---|---|
| ASSIGN_PROCESS_TO_JOB | Specification for whether or not UCMD Server assigns child processes to a single Windows job object. |
| COMMAND_FILE_ENCRYPTED | Encrypted command file. |
| COMMAND_FILE_PLAIN | Pain text command file. |
| ENCRYPTION_KEY | Encryption key used to decrypt an encrypted command file specified by option COMMAND_FILE_ENCRYPTED. |
| SERVER_OPTIONS | Universal Command Server options that can be overridden by Managers. |

## Remote Category Options

| Option Name | Description |
|---|---|
| CONNECT_TIMEOUT | Amount of time that a UCMD Manager will wait for a connection to a remote Universal Broker to complete. |
| DNS_EXPAND | Number of IP addresses returned to UCMD Manager following a DNS query issued to resolve a host name. |
| HOST_SELECTION | Host in the REMOTE_HOST list that the UCMD Manager will choose to begin its attempts to connect to a remote Universal Broker. |
| HOSTNAME_RETRY_COUNT | Number of times that UCMD will attempt to resolve the host name of a specified Universal Broker before it ends with a connect error. |
| MFT_SAFE_MODE | Situations in which more than one host may be specified in the REMOTE_HOST list when manager fault tolerance (MFT) is enabled. |
| OUTBOUND_IP | Host or IP address to use for all outgoing IP connections. |
| REMOTE_HOST | List of one or more hosts upon which a command may run. |
| REMOTE_PORT | TCP/IP port number of the remote Broker. |

## Standard File Category Options

| Option Name | Description |
|---|---|
| SIO_DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on a standard file. |
| SIO_DATA_COMPRESSION | Specification for whether or not data is compressed on a standard file (and if so, how). |
| SIO_DATA_ENCRYPTION | Specification for whether or not data is encrypted on a standard file. |
| SIO_LOCAL_CODE_PAGE | Code page used for local text translation on a standard file. |
| SIO_LOCAL_FILE | Local file used for a standard file instead of the default. |
| SIO_MODE | Translation mode of a standard file. |
| SIO_REMOTE_CODE_PAGE | Code page used for remote text translation on a standard file. |
| STDERR_FILE_SPEC | Start of standard error file specification options. |
| STDIN_FILE_SPEC | Start of standard input file specification options. |
| STDOUT_FILE_SPEC | Start of standard output file specification options. |

## User Category Options

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

## 4.2.4 Command Line Syntax

Figure 4.1 and Figure 4.2, below, illustrate the command line syntax – using the command line, long form of the configuration options – of UCMD Manager for Windows.

```
ucmd
{ -cmd command [-cmd_type {cmdref|shell} ] | -script file [-options options]
   [-script_type type] }
-host hostlist
[-connect_timeout seconds]
[-dns_expand {yes|no} ]
[-host_selection {sequential|random} ]
[-mft_safe_mode {yes|no} ]
[-file filename | -encryptedfile filename [-key key] ]  *
[-port port]
[-userid user [-pwd pwd] ]
[-hostname_retry_count count]
[-outboundip host]
[-server options]
[-assign_process_to_job option]
[-managerft {yes|no} ]
[-cmdid id]
[-login {yes|no} ]
[-lang language]
[-level {trace|audit|info|warn|error}[,{time|notime} ]
[-tracefilelines lines]
[-trace_table size,{error|always|never} ]
[-ca_certs file [-verify_host_name {yes|no|hostname} ]
   [-verify_serial_number number] ]
[-cert file -private_key file [-private_key_pwd password] ]
[-crl file]
[-ctl_ssl_cipher_list cipherlist]
[-data_ssl_cipher_list cipherlist]
[-default_cipher cipher]
[-forcecomplete {yes|no} ]
[-job_retention seconds]
[-delay seconds]
[-networkft {yes|no} ]
[-retry_count number]
[-retry_interval seconds]
[-restart {yes|no|auto} [-managerft {yes|no} [-cmdid id] ] ]
[-codepage codepage]
[-compress {yes|no}[,{zlib|hasp} ] ]
```

Figure 4.1 Universal Command for Windows - Command Line Syntax (1 of 2)

```
[-encrypt {yes|no} ]
[-authenticate {yes|no} ]
[-stdin | -stdout | -stderr]
   [-codepage codepage]
   [-compress {yes|no}[,{zlib|hasp} ] ]
   [-encrypt {yes|no} ]
   [-authenticate {yes|no} ]
   [-localfile ddname]
   [-mode {text|binary}[,{ucs|direct} ]
   [-remotecodepage codepage]
[-exit_code_map map]
[-comment text]

ucmd
{-help | -version}
```

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

Figure 4.2 Universal Command for Windows - Command Line Syntax (2 of 2)

# 4.3 Examples of UCMD Manager for Windows

Appendix A Examples provides operating system-specific examples that demonstrate the use of Universal Command.

Included in this appendix are the following examples that demonstrate the use of Universal Command Manager for Windows:

- File Copy Example
- System Status Report Example
- Directory Backup Example
- Directory Restore Example
- Manager Command
- Command Coded as a Script
- Redirect Standard Out and Standard Error
- Spawn Background Process with nohup: UNIX
- Redirect Standard Input from Initiating System
- Redirect Standard Input from /dev/null
- Authentication Parameters from Encrypted File
- Manager Fault Tolerance

# 4.4 Security

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

UCMD security concerns are:

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

## 4.4.1 File Permissions

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

Eligible users of UCMD require read access to the message catalogs (`*.umc` files) in the `nls` subdirectory of the UCMD Manager installation directory. If UCMD Manager is installed on an NTFS partition, these file permissions are set automatically during the installation.

## 4.4.2 Universal Configuration Manager

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

# Chapter 5
# Universal Command Manager for UNIX

## 5.1 Overview

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

UCMD Manager provides a command line interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered the command directly on the remote command line.

On the command line, you must specify a command to execute and a remote Universal Broker. Additional input to each execution of the UCMD Manager command is made via configuration options, which control product behavior and resource allocation for that execution.

Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

UCMD Manager registers with a locally running Universal Broker. Consequentially, a Universal Broker must be running in order for a UCMD Manager to execute.

# 5.2 Usage

This section describes the command input, configuration and configuration options, and command line syntax of UCMD Manager for UNIX.

Section 5.3 Examples of UCMD Manager for UNIX provides some examples that demonstrate the flexibility of Universal Command.

## 5.2.1 Standard Input

The UCMD Manager command is executed from an interactive UNIX shell or as a shell script. The **ucmd** command reads from standard input and writes it to the UCMD Server for the remote command to read as its standard input.

When the UCMD Manager is executed from an interactive shell, standard input is allocated to the terminal. Any characters typed in the terminal are read as standard input by ucmd and transmitted to the UCMD Server. If **ucmd** is executing a remote command that is reading standard input, it will read the characters being typed in the terminal until it receives an end-of-file indicator. To enter end-of-file in an interactive shell, press <Ctrl+D> at the start of a new line.

The allocation of standard input can be changed with a shell redirection operator. The redirection operators instruct the shell to change the allocation of the standard files. To change the allocation of standard input, use the **<** operator. The shell can redirect to a special file referred to as **/dev/null**. The **/dev/null** file is always empty if read from and never full if written to (all data written to **/dev/null** is never saved on disk or in memory). To allocate standard input to **/dev/null** the command syntax is as follows:

```
ucmd [OPTIONS...] < /dev/null
```

If **ucmd** is executed as a background job (using the **&** operator), it will receive the SIGTTIN signal when **ucmd** tries to read from standard input. Background jobs cannot read their standard input from the terminal since the foreground job (or the shell) has it allocated. The **ucmd** job is stopped until it is brought to the foreground.

To run a **ucmd** job that does not require terminal input in the background, redirect its standard input from **/dev/null**.

## 5.2.2 Configuration

Configuration consists of:

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

UCMD Manager for UNIX 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, **`ucmd.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 Universal Products configuration methods and command input for Universal Products.

# 5.2.3 Configuration Options

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

## Configuration Options Categories

Table 5.1, below, categorizes the configuration options into logical areas of application.

| Category | Description |
|---|---|
| Certificates | X.509 certificate related options. |
| Command | Command or script to execute on the remote system. If a script is being executed, the script may reside on the local host on which the Manager is running or the remote host on which the Server is running.<br>It also includes options to control the process environment in which the command executes. |
| 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 Command message options. |
| Miscellaneous | Options use to display command help and program versions. |
| Network | Processing options for all the data transferred between the remote and local systems. |
| Options | Alternative methods to specify command options. |
| Remote | Network address of the remote system and connection options. |
| User | User account the command executes with on the remote system. |
| Standard File | Processing options for the standard files transferred between the remote and local systems.<br>The STDFILE options are specified differently then the other options. There are three types of standard files: stderr, stdin, and stdout. Each standard file can have a different set of options applied. In order to distinguish between the standard files, the options must start with a standard file specification option (STDERR_SPEC, STDIN_SPEC, or STDOUT_SPEC). The standard file options (names starting with SIO_) follow the standard file specification option. (Section 5.3 Examples of UCMD Manager for UNIX demonstrates this.) |

Table 5.1 Universal Command Manager for UNIX - Configuration Options Categories

The UCMD Manager for UNIX 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 Command 3.2.0 Reference Guide.

## Certificate Category Options

| Option Name | Description |
|---|---|
| CA_CERTIFICATES | Location of 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 for whether or not the Broker's X.509 certificate host name field must be verified. |
| VERIFY_SERIAL_NUMBER | Specification for whether or not the Broker's X.509 certificate serial number field must be verified. |

## Command Category Options

| Option Name | Description |
|---|---|
| COMMAND | Remote command to execute. |
| COMMAND_ID | Unique command ID associated the unit of work. |
| COMMAND_TYPE | Type of command specified with option COMMAND. |
| EXIT_CODE_MAP | Allows exit codes from the user process executed by UCMD Server to be translated (mapped) to a corresponding exit code for UCMD Manager. |
| LOGIN | Specification for whether or not the command runs in a login environment. |
| MANAGER_FAULT_TOLERANT | Specification for whether or not manager fault tolerance is used. |
| SCRIPT_FILE | Local script file to execute on the remote system. |
| SCRIPT_OPTIONS | Command line options passed to the script file. |
| SCRIPT_TYPE | Type of script file specified by option SCRIPT_FILE. |

## 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 UCMD Manager 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 |
| TRACE_FILE_LINES | Maximum number of lines written to a trace file before it wraps around. |
| TRACE_TABLE | Memory trace table specification. |

## Miscellaneous Category Options

| Option Name | Description |
|---|---|
| COMMENT | User-defined string. |
| HELP | Write command option help. |
| VERSION | Write program version. |

## Network Category Options

| Option Name | Description |
|---|---|
| CODE_PAGE | Code page used for text translation. |
| CTL_SSL_CIPHER_LIST | SSL cipher list for the control session. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the data sessions. |
| DEFAULT_CIPHER | Default SSL cipher used for data sessions. |
| FORCE_COMPLETE | Forces a manager fault tolerant server in a PENDING communication state to COMPLETED state without retrieving the spooled data. |
| JOB_RETENTION | Length of time that a Server waits for a reconnect after the user process completes. |
| NETWORK_DELAY | Maximum number of seconds considered acceptable to wait for data communications. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not the network fault tolerant protocol is used. |
| RECONNECT_RETRY_COUNT | Maximum number of network fault tolerant reconnect attempts. |
| RECONNECT_RETRY_INTERVAL | Number of seconds between network fault tolerant reconnect attempts. |
| RESTART | Specification for whether or not the manager is requesting restart. |

## Options Category Options

| Option Name | Description |
|---|---|
| ASSIGN_PROCESS_TO_JOB | Specification for whether or not UCMD Server assigns child processes to a single Windows job object. |
| 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. |
| SERVER_OPTIONS | UCMD Server options that can be overridden by UCMD Managers. |

## Remote Category Options

| Option Name | Description |
|---|---|
| CONNECT_TIMEOUT | Amount of time that a UCMD Manager will wait for a connection to a remote Universal Broker to complete. |
| DNS_EXPAND | Number of IP addresses returned to UCMD Manager following a DNS query issued to resolve a host name. |
| HOST_SELECTION | Host in the REMOTE_HOST list that the UCMD Manager will choose to begin its attempts to connect to a remote Universal Broker. |
| HOSTNAME_RETRY_COUNT | Number of times that UCMD will attempt to resolve the host name of a specified Universal Broker before it ends with a connect error. |
| MFT_SAFE_MODE | Situations in which more than one host may be specified in the REMOTE_HOST list when manager fault tolerance (MFT) is enabled. |
| OUTBOUND_IP | Host or IP address to use for all outgoing IP connections. |
| REMOTE_HOST | List of one or more hosts upon which a command may run. |
| REMOTE_PORT | TCP/IP port number of the remote Broker. |

## Standard File Category Options

| Option Name | Description |
|---|---|
| SIO_DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on a standard file. |
| SIO_DATA_COMPRESSION | Specification for whether or not data is compressed on a standard file (and if so, how). |
| SIO_DATA_ENCRYPTION | Specification for whether or not data is encrypted on a standard file. |
| SIO_LOCAL_CODE_PAGE | Code page used for local text translation on a standard file. |
| SIO_LOCAL_FILE | Local file used for a standard file instead of the default. |
| SIO_MODE | Translation mode of a standard file. |
| SIO_REMOTE_CODE_PAGE | Code page used for remote text translation on a standard file. |
| STDERR_FILE_SPEC | Start of standard error file specification options. |
| STDIN_FILE_SPEC | Start of standard input file specification options. |
| STDOUT_FILE_SPEC | Start of standard output file specification options. |

## User Category Options

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

## 5.2.4 Command Line Syntax

Figure 5.1 and Figure 5.2, below, illustrate the command line syntax – using the command line, long form of the configuration options – of UCMD Manager for UNIX.

```
ucmd
{ -cmd command [-cmd_type {cmdref|shell} ] | -script file [-options options]
   [-script_type type] }
-host hostlist
[-connect_timeout seconds]
[-dns_expand {yes|no} ]
[-host_selection {sequential|random} ]
[-mft_safe_mode {yes|no} ]
[-file filename | -encryptedfile filename [-key key] ]  *
[-port port]
[-userid user [-pwd pwd] ]
[-hostname_retry_count count]
[-outboundip host]
[-bif_directory directory]
[-plf_directory directory]
[-server options]
[-assign_process_to_job option]
[-managerft {yes|no} ]
[-cmdid id]
[-login {yes|no} ]
[-lang language]
[-level {trace|audit|info|warn|error}[,{time|notime} ]
[-tracefilelines lines]
[-trace_table size,{error|always|never} ]
[-ca_certs file [-verify_host_name {yes|no|hostname} ]
   [-verify_serial_number number] ]
[-cert file -private_key file [-private_key_pwd password] ]
[-crl file]
[-ctl_ssl_cipher_list cipherlist]
[-data_ssl_cipher_list cipherlist]
[-default_cipher cipher]
[-forcecomplete {yes|no} ]
[-job_retention seconds]
[-delay seconds]
[-networkft {yes|no} ]
[-retry_count number]
[-retry_interval seconds]
```

Figure 5.1 Universal Command for UNIX - Command Line Syntax (1 of 2)

```
[-restart {yes|no|auto} [-managerft {yes|no} [-cmdid id] ] ]
[-codepage codepage]
[-compress {yes|no}[,{zlib|hasp} ] ]
[-encrypt {yes|no} ]
[-authenticate {yes|no} ]
[-stdin | -stdout | -stderr]
   [-codepage codepage]
   [-compress {yes|no}[,{zlib|hasp} ] ]
   [-encrypt {yes|no} ]
   [-authenticate {yes|no} ]
   [-localfile ddname]
   [-mode {text|binary}[,{ucs|direct} ] ]
   [-remotecodepage codepage]
[-exit_code_map map]
[-comment text]

ucmd
{-help | -version}
```

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

Figure 5.2 Universal Command for UNIX - Command Line Syntax (2 of 2)

# 5.3 Examples of UCMD Manager for UNIX

Appendix A Examples provides operating system-specific examples that demonstrate the use of Universal Command.

Included in this appendix are the following examples that demonstrate the use of Universal Command Manager for UNIX:

- File Copy Example 1
- File Copy Example 2
- Network Status Script Example
- Manager Command
- Command Coded as a Script
- Redirect Standard Out and Standard Error
- Redirect Standard Input from Initiating System
- Redirect Standard Input from /dev/null
- Authentication Parameters from Encrypted File
- Manager Fault Tolerance
-

# 5.4 Security

Universal Command 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 Command Manager has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

Universal Command security concerns are:

1. Access to Universal Command files and directories.
2. Access to Universal Command configuration options.
3. The privacy and integrity of transmitted network data.

## 5.4.1 File Permissions

Only trusted user accounts should have write permission to the Universal Command Server installation directory, subdirectories, and all files within. This most likely means only an administration group should have write access. Eligible users of Universal Command require read access to the message catalogs (`*.umc` files) in the `nls` subdirectory of the Universal Command Manager installation directory.

## 5.4.2 Configuration Files

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

# Chapter 6
# Universal Command Manager for OS/400

## 6.1 Overview

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

UCMD Manager for OS/400 provides a OS/400 command interface to remote computers running the UCMD Server component. UCMD Manager executes remote commands as they would be if you entered the command directly on the remote command line.

On the command line, you must specify a command to execute and a remote Universal Broker. Additional input to each execution of the UCMD Manager command is made via configuration options, which control product behavior and resource allocation for that execution.

Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

# 6.2 Usage

UCMD Manager for OS/400 executes as a CL command. This section describes the command environment, configuration and configuration options, and command line syntax. Section 6.3 Examples of UCMD Manager for OS/400 provides some examples that demonstrate the flexibility of Universal Command.

## 6.2.1 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 UCMD, substitute the tagged names for the untagged names in these references.

## 6.2.2 Command Execution Environment

The command is valid in all environments:

1. Batch input streams
2. CL programs
3. REXX procedures
4. CL ILE modules
5. Interactive processing
6. Passed to the system program QCMDEXC (or QCAEXEC) for processing

## 6.2.3 Configuration

Configuration consists of:

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

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

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

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

The configuration file, **`UNVPRD320/UNVCONF(UCMD)`**, 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 Universal Products configuration methods and command input for Universal Products.

# 6.2.4 Configuration Options

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

## Configuration Options Categories

Table 6.1, below, categorizes the configuration options into logical areas of application.

| Category | Description |
|---|---|
| Certificates | X.509 certificate related options. |
| Command | Command or script to execute on the remote system. If a script is being executed, the script may reside on the local host on which the Manager is running or the remote host on which the Server is running.<br>It also includes options to control the process environment in which the command executes. |
| Events | Options used to define event generation. |
| Local | Options required for local broker registration. |
| Messages | Universal Command message options. |
| Miscellaneous | Options use to display command help and program versions. |
| Network | Processing options for all the data transferred between the remote and local systems. |
| Options | Alternative methods to specify command options. |
| Remote | Network address of the remote system and connection options. |
| Standard File | Processing options for the standard files transferred between the remote and local systems.<br>At the program interface level, the STDFILE options are specified differently then the other options. There are three types of standard files: stdin, stdout, and stderr. Each standard file can have a different set of options applied.<br>In order to distinguish between the standard files, the options must start with a standard file specification option (STDERR_FILE_SPEC, STDIN_FILE_SPEC, or STDOUT_FILE_SPEC). The standard file options (see Standard File Category Options) follow the standard file specification option. The STRUCM command interface hides this difference from the user. |
| User | User account the command executes with on the remote system. |

Table 6.1 Universal Command Manager for OS/400 - Command Options Categories

The UCMD Manager 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 Command 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 for whether or not the Broker's X.509 certificate host name field must be verified. |
| VERIFY_SERIAL_NUMBER | Specification for whether or not the Broker's X.509 certificate serial number field must be verified. |

## Command Category Options

| Option Name | Description |
|---|---|
| COMMAND | Remote command to execute. |
| COMMAND_ID | Unique command ID associated the unit of work. |
| COMMAND_TYPE | Type of command specified with option COMMAND. |
| EXIT_CODE_MAP | Allows exit codes from the user process executed by UCMD Server to be translated (mapped) to a corresponding exit code for UCMD Manager. |
| LOGIN | Specification for whether or not the command runs in a login environment. |
| MANAGER_FAULT_TOLERANT | Specification for whether or not manager fault tolerance is used. |
| SCRIPT_FILE | Local script file to execute on the remote system. |
| SCRIPT_OPTIONS | Command line options passed to the script file. |
| SCRIPT_TYPE | Type of script file specified by option SCRIPT_FILE. |

## 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. |
| TRACE_FILE_LINES | Maximum number of lines written to a trace file before it wraps around. |
| TRACE_TABLE | Memory trace table specification. |

## Miscellaneous Category Options

| Option Name | Description |
|---|---|
| COMMENT | User-defined string. |
| VERSION | Write program version. |

## Network Category Options

| Option Name | Description |
|---|---|
| CODE_PAGE | Code page used for text translation. |
| CTL_SSL_CIPHER_LIST | SSL cipher list for the control session. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the data sessions. |
| DEFAULT_CIPHER | Default SSL cipher used for data sessions. |
| FORCE_COMPLETE | Forces a manager fault tolerant server in a PENDING communication state to COMPLETED state without retrieving the spooled data. |
| JOB_RETENTION | Length of time that a Server waits for a reconnect after the user process completes. |
| NETWORK_DELAY | Maximum number of seconds considered acceptable to wait for data communications. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not the network fault tolerant protocol is used. |
| RECONNECT_RETRY_COUNT | Maximum number of network fault tolerant reconnect attempts. |
| RECONNECT_RETRY_INTERVAL | Number of seconds between network fault tolerant reconnect attempts. |
| RESTART | Specification for whether or not the manager is requesting restart. |

## Options Category Options

| Option Name | Description |
|---|---|
| ASSIGN_PROCESS_TO_JOB | Specification for whether or not UCMD Server assigns child processes to a single Windows job object. |
| 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. |
| SERVER_OPTIONS | UCMD Server options that can be overridden by UCMD Managers. |

## Remote Category Options

| Option Name | Description |
|---|---|
| CONNECT_TIMEOUT | Amount of time that a UCMD Manager will wait for a connection to a remote Universal Broker to complete. |
| DNS_EXPAND | Number of IP addresses returned to UCMD Manager following a DNS query issued to resolve a host name. |
| HOST_SELECTION | Host in the REMOTE_HOST list that the UCMD Manager will choose to begin its attempts to connect to a remote Universal Broker. |
| HOSTNAME_RETRY_COUNT | Number of times that UCMD will attempt to resolve the host name of a specified Universal Broker before it ends with a connect error. |
| MFT_SAFE_MODE | Situations in which more than one host may be specified in the REMOTE_HOST list when manager fault tolerance (MFT) is enabled. |
| 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. |

## Standard File Category Options

| Option Name | Description |
|---|---|
| SIO_LOCAL_CODE_PAGE | Code page used for local text translation on a standard file. |
| SIO_LOCAL_FILE | Local file used for a standard file instead of the default. |
| SIO_MODE | Translation mode of a standard file. |
| SIO_REMOTE_CODE_PAGE | Code page used for remote text translation on a standard file. |

## User Category Options

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

## 6.2.5 Command Line Syntax

Figure 6.1 and Figure 6.2, below, illustrate the command line syntax — using the STRUCM parameter form of the configuration options — of UCMD Manager for OS/400.

After the positional options, which appear immediately after the STRUCM command, the options are organized by category, as identified in Section 6.2.4 Configuration Options.

```
STRUCM
{ CMD(command) [CMDTYPE({cmd|cmdref|rexx})] | SCRFILE(file) [SCRMBR(member)]
   [OPTIONS(options)] [SCRTYPE(type)] }
HOST(hostaddress)
[PORT(port)]
[USERID(user) [PWD(pwd)] ]
REMOTE CATEGORY:
[CONNECTTO(seconds)]
[DNSEXPAND(*option)]
[HOSTSELECT(*option)]
[HSTNMRTYCT(count)]
[MFTSAFMODE(*option)]
[OUTBOUNDIP(host)]
OPTIONS CATEGORY:
[ASSIGNPROC(*options)]
[CMDFILE(filename) [CMDMBR(member)] ] | [ECMFILE(filename) [ECMMBR(member)]
   [KEY(key)] ]
[SERVER(options)]
COMMAND CATEGORY:
[CMDID(id)]
[EXITCDMAP(option)]
[LOGIN({yes|no})]
[MANAGERFT({yes|no})]
MESSAGES CATEGORY:
[MSGLANG(language)]
[MSGLEVEL(*{trace|audit|info|warn|error})]      NOTE: Value trace turns trace on.
[TRCLINES(lines)]
[TRCTBL(size,{error|always|never}) ]
[CACERTS(file [lib] ) [CACERTSMBR(member)] [VFYHSTNM({yes|no|hostname})]
   [VFYSERNUM(number)] ]
[CERT(file [lib] ) [CERTMBR(member)]
     PVTKEYF(file [lib] ) [PVTKEYFMBR(member)] [PVTKEYPWD(password)] ]
[CRLFILE(file [lib]) [CRLMBR(member)] ]
```

Figure 6.1 Universal Command Manager for OS/400 - Command Line Syntax (1 of 2)

**NETWORK CATEGORY:**
[AUTH(*{**yes**|**no**})]
[CODEPAGE(*codepage*)]
[COMPRESS(*{**yes**|**no**}[CMPRSMTH*{**zlib**|**hasp**}] ) ]
[CTLCPHRLST(*cipherlist*)]
[DELAY(*seconds*)]
[DFTCPHR(*cipher*)]
[DTACPHRLST(*cipherlist*)]
[ENCRYPT(*{**yes**|**no**})]
[FRCCMPLT({**yes**|**no**})]
[JOBRTN(*seconds*)]
[NETWORKFT({**yes**|**no**})]
[RESTART({**yes**|**no**|**auto**}) [MANAGERFT({**yes**|**no**}) [CMDID(*id*)] ] ]
[RETRYCNT(*number*)]
[RETRYINT(*seconds*)]
[SERFILE(*output_file*)]
[SERMBR(*member*)]
[SERMODE(*{**text**|**binary**})]
[SERCPG(*codepage*)]
[SERRCP(*codepage*)]
[SINFILE(*input_file*)]
[SINMBR(*member*)]
[SINMODE(*{**text**|**binary**})]
[SINCPG(*codepage*)]
[SINRCP(*codepage*)]
[SOTCPG(*codepage*)]
[SOTFILE(*output_file*)]
[SOTMBR(*member*)]
[SOTMODE(*{**text**|**binary**})]
[SOTRCP(*codepage*)]
**LOCAL CATEGORY:**
[PLFDIR(*directory*)]
**MISCELLANEOUS CATEGORY:**
[COMMENT(*user-defined string*)]

**STRUCM**
VERSION(*{**yes**|**no**})

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

Figure 6.2 Universal Command Manager for OS/400 - Command Line Syntax (2 of 2)

# 6.3 Examples of UCMD Manager for OS/400

Appendix A Examples provides operating system-specific examples that demonstrate the use of Universal Command.

Included in this appendix are the following examples that demonstrate the use of Universal Command Manager for OS/400:

- File Copy Example 1
- File Copy Example 2
- File Copy Example 3
- File Copy Example 4
- Display Library with Manager Fault Tolerance Active Using USBMJOB Example
- Network Status Script Example
- Manager Command
- Copy File from Remote Windows to Local OS/400
- Command Coded as a Script

These examples reference the OS/400 commands by their untagged names. If you are using commands with tagged names to run UCMD, substitute the tagged names for the untagged names. (See Section 6.2.1 Universal Products for OS/400 Commands for further information.)

# 6.4 Security

Universal Command 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. UCMD Manager has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

UCMD security concerns are:

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

## 6.4.1 File Permissions

Only administrator accounts and the **UNVUBR320** user profile should have write permission to the UCMD Server product library (**UNVPRD320**) and files within. Eligible users of UCMD require read access to the message catalogs in the **UNVNLS** file.

## 6.4.2 Configuration Files

Only administrator accounts should have write access to UCMD's configuration file.

# Chapter 7
# Universal Command Manager for HP NonStop

## 7.1 Overview

This chapter documents the Universal Command (UCMD) Manager at a detailed level, specific to the HP NonStop variety of operating systems.

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

A Universal Command (UCMD) Manager executes commands on any computer running the UCMD Server component. You indicate to the UCMD Manager what command(s) to execute and how the standard input and output data should be processed. The UCMD Manager connects to the UCMD Server and processes your request.

The UCMD Manager for HP NonStop provides a command line interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered them directly on the remote command line. Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

# 7.2 Usage

UCMD Manager for HP NonStop executes as a command line application. This section describes the command input, configuration and configuration options, and command line syntax. Section 7.3 Examples of UCMD Manager for HP NonStop provides some examples demonstrating the flexibility of Universal Command.

## 7.2.1 Standard Input

The UCMD Manager command, **ucmd**, is executed from an interactive TACL shell or as a shell script. **ucmd** reads from standard input and writes it to the UCMD Server for the remote command to read as its standard input. When the UCMD Manager is executed from an interactive shell, standard input is allocated to the terminal. Any characters typed in the terminal are read as standard input by ucmd and transmitted to the UCMD Server.

If **ucmd** is executing a remote command that is reading standard input, it will read the characters being typed in the terminal until it receives an end-of-file indicator. To enter end-of-file in an interactive shell, press <Ctrl+Y> at the start of a new line.

## 7.2.2 Configuration

Configuration consists of:

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

UCMD Manager for HP NonStop 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, **ucmdcfg**, 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 Universal Products configuration methods.

# 7.2.3 Configuration Options

UCMD Manager for HP NonStop consists of a command line program followed by a list of configuration options. The UCMD Manager has many different options that specify what is executed, how it is executed, how network data is processed, and much more.

## Configuration Option Categories

Table 7.1, below, categorizes the configuration options into logical areas of application.

| Category | Description |
|---|---|
| Command | Command or script to execute on the remote system. If a script is being executed, the script may reside on the local host on which the UCMD Manager is running or the remote host on which the UCMD Server is running.<br>The Command category also includes options to control the process environment in which the command executes. |
| Messages | Universal Command message options. |
| Miscellaneous | Options use to display command help and program versions. |
| Network | Processing options for all the data transferred between the remote and local systems. |
| Options | Alternative methods to specify command options. |
| Remote | Network address of the remote system and connection options. |
| Standard File | Processing options for the standard files transferred between the remote and local systems.<br>The STDFILE options are specified differently then the other options. There are three types of standard files: stderr, stdin, and stdout. Each standard file can have a different set of options applied. In order to distinguish between the standard files, the options must start with a standard file specification option (STDERR_SPEC, STDIN_SPEC, or STDOUT_SPEC). The standard file options (names starting with SIO_) follow the standard file specification option. Section 7.3 Examples of UCMD Manager for HP NonStop demonstrates this. |
| User | User account the command executes with on the remote system. |

Table 7.1 Universal Command Manager for HP NonStop - Command Option Categories

The UCMD Manager for HP NonStop 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 Command 3.2.0 Reference Guide.

## Command Category Options

| Option Name | Description |
|---|---|
| COMMAND | Remote command to execute. |
| COMMAND_ID | Unique command ID associated the unit of work. |
| LOGIN | Specification for whether or not the command runs in a login environment. |
| SCRIPT_FILE | Local script file to execute on the remote system. |
| SCRIPT_OPTIONS | Command line options passed to the script file. |
| SCRIPT_TYPE | Type of script file specified by option SCRIPT_FILE. |

## 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 for text translation. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| JOB_RETENTION | Length of time that a UCMD Server waits for a reconnect after the user process completes. |
| NETWORK_DELAY | Maximum number of seconds considered acceptable to wait for data communications. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not network fault tolerant protocol is used. |
| RECONNECT_RETRY_COUNT | Maximum number of network fault tolerant reconnect attempts. |
| RECONNECT_RETRY_INTERVAL | Length of time between network fault tolerant reconnect attempts. |

## 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. |
| SERVER_OPTIONS | UCMD Server options that can be overridden by UCMD Managers. |

## 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. |

## Standard File Category Options

| Option Name | Description |
|---|---|
| SIO_DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on a standard file. |
| SIO_DATA_COMPRESSION | Specification for whether or not data is compressed on a standard file (and if so, how). |
| SIO_DATA_ENCRYPTION | Specification for whether or not data is encrypted on a standard file. |
| SIO_LOCAL_CODE_PAGE | Code page used for local text translation on a standard file. |
| SIO_LOCAL_FILE | Local file used for a standard file instead of default. |
| SIO_MODE | Translation mode of a standard file. |
| SIO_REMOTE_CODE_PAGE | Code page used for remote text translation on a standard file. |
| STDERR_FILE_SPEC | Start of standard error file specification options. |
| STDIN_FILE_SPEC | Start of standard input file specification options. |
| STDOUT_FILE_SPEC | Start of standard output file specification options. |

## User Category Options

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

## 7.2.4 Command Line Syntax

Figure 7.1 , below, illustrates the command line syntax — using the command line, long form of the configuration options — of UCMD Manager for HP NonStop.

```
ucmd
{ -cmd command | -script file [-options options] [-script_type type] }
-host hostaddress
[-file filename | -encryptedfile ddname [-key key] ]  *
[-port port]
[-userid user [-pwd pwd] ]
[-server options]
[-managerft {yes|no} ]
[-cmdid id]
[-login {yes|no} ]
[-lang language]
[-level {trace|audit|info|warn|error}[,{time|notime} ]
[-verify_host_name {yes|no|hostname} ]
[-job_retention seconds]
[-delay seconds]
[-networkft {yes|no} ]
[-retry_count number]
[-retry_interval seconds]
[-codepage codepage]
[-compress {yes|no}[,{zlib|hasp} ] ]
[-encrypt {yes|no} ]
[-authenticate {yes|no} ]
[-stdin | -stdout | -stderr]
   [-codepage codepage]
   [-compress {yes|no}[,{zlib|hasp} ] ]
   [-encrypt {yes|no} ]
   [-authenticate {yes|no} ]
   [-localfile ddname]
   [-mode {text|binary}[,{ucs|direct} ] ]
   [-remotecodepage codepage]

ucmd
{-help | -version}
```

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

Figure 7.1 UCMD Manager for HP NonStop - Command Line Syntax

# 7.3 Examples of UCMD Manager for HP NonStop

Appendix A Examples provides operating system-specific examples that demonstrate the use of Universal Command.

Included in this appendix are the following examples that demonstrate the use of Universal Command Manager for HP NonStop:

- File Copy Example 1
- File Copy Example 2
- Network Status Report Example
- Manager Command
- Copy Remote Windows File to Local HP NonStop
- Copy Local File to Remote Windows
- Command Coded as a Script

# 7.4 Security

UCMD 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. UCMD Manager has balanced the two to avoid the administrative complexity with a minimum sacrifice to security.

UCMD security concerns are:

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

## 7.4.1 File Permissions

Only trusted user accounts should have write permission to the UCMD Server installation subvolumes and all of the files within them. This most likely means only an administration group should have write access. Eligible users of UUCMD require read access to the message catalogs in the **$SYSTEM.UNVNLS** subvolume.

## 7.4.2 Configuration Files

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

# Chapter 8
# Universal Command Server for z/OS

## 8.1 Overview

This chapter provides information on the Universal Command (UCMD) Server specific to the z/OS operating system.

The UCMD Server is responsible for:

1. Accepting a request from a UCMD Manager
2. Establishing network connections with the Manager
3. Creating another process to execute the Manager's command
4. Transmitting output and input files between the Server and Manager
5. Returning the command's exit code to the Manager

The Broker, at the Manager's request, starts the Server. The Server processes the request from the Manager and then exits. A new Server process is created for each Manager request. There may be zero or more Servers running simultaneously, processing requests from different Managers.

### 8.1.1 Environment

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

All components dependent upon Universal Broker, such as UCMD Server, inherit the message language from Universal Broker. All messages issued by components are sent to the Universal Broker for writing.

# 8.2 Commands

There are three types of work that a z/OS Universal Command Server can execute:

1. z/OS USS commands and scripts
2. Started Tasks
3. Command References

In all cases, the work executes in its own address space with its own user identity. No Universal Products programs share the address space with the unit of work started by the Server.

## 8.2.1 z/OS UNIX System Services Command

The UCMD Server's default command type is the z/OS USS shell. This can be customized with the COMMAND_TYPE configuration option. USS shell commands are executed in a USS process within its own address space.

A UCMD Manager requests the execution of a USS command by specifying a COMMAND_TYPE of **`shell`**. USS scripts are requested by specifying a SCRIPT_TYPE of **`shell`**.

The environmental attributes of the user process are described in the following sections.

### User Identification

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

- With user security active, the UCMD Server requires the UCMD Manager to supply a valid z/OS user ID and a password. The user process executes with the user ID and the primary and secondary group ID's of the user. The user profile must have a properly defined OMVS segment.
- With user security inactive, the Server does not require the Manager to supply a valid user ID. The user process executes with the user ID of the Server. The Server inherits its user ID from the Broker started task, which is a superuser account (UID 0).

  The superuser account provides a lot of access to the operating system that a user process typically does not require. Setting security inactive is not recommended because of the level of access it permits the user process.

## Working Directory

The working directory of a user process depends on whether user security is active or inactive:

- With user security active, a user process's working directory is the home directory of the user ID as defined in the user profile's OMVS segment HOME parameter value.
- With user security inactive, a user process's working directory is the working directory as defined by the Universal Broker's user profile OMVS segment HOME parameter value. All user processes executed will use the same directory. Care should be taken to avoid name clashes and other consequences of multiple processes sharing a working directory.

## Command Shell

The UCMD Manager LOGIN option and the UCMD Server LOGIN option determine what command shell is used.

For non-login environments, the default is shell **`/bin/sh`**. The shell used for non-login environments is configurable with the SHELL option.

For login environments, the shell defined in the user ID's OMVS segment with the SHELL option is used. The shell environment is created as if the user logged on interactively. For example, the shell's **`.profile`** is used to initialize the environment.

The non-login environment is similar to the environment that the **`cron`** scheduler provides. User resource files, such as **`.profile`**, are not utilized.

The application scripts being executed and your local system management policies should be used to determine which method is best.

If user security is inactive, the default shell **`/bin/sh`** always is used independent of the LOGIN option.

## Environment Variables

Environment variables are inherited from the Universal Command Server, which in turn inherits them from the Universal Broker. If security is active, certain variables are modified to match the user environment. They are HOME, LOGNAME, USER, PWD, and SHELL. Their values are update to reflect the values for the new environment.

The following variables are added if not found in the environment: HOME, USER, SHELL, and UCMDENV. The UCMDENV variable is set to a value of *1*. It can be used within scripts to determine if Universal Command has invoked them.

The UCMD Manager LOGIN option and the UCMD Server LOGIN option have an impact on the environment variables defined. For login environments, the user's shell is invoked as a login shell, which, in turn, uses the shell profile file in the user's home directory. So any environment variables set in the profile file also will be defined.

If user security is inactive, no changes are made to the environment variables.

# 8.2.2 Started Tasks

The Universal Command Server has the ability to execute z/OS started tasks. Started tasks have some advantages over USS commands. They execute z/OS programs using standard JCL. The JCL must be predefined in a system procedure library.

UCMD Managers refer to the started task by name and optionally provide an input file and JCL overrides. A Manager requests the execution of a started task by specifying a COMMAND_TYPE of **`stc`**.

Started task requests are processed by the Universal Command Server Command Processor for Started Tasks (UCMSCPST). The Command Processor (CP) is executed by the Server as a USS process within its own address space.

The STC CP execution environment is the same as the USS command environment described in Section 8.2.1 z/OS UNIX System Services Command.

## Extended MCS Console

The started task is started with the START system command through an extended MCS console. Refer to the IBM *MVS System Commands* manual for a complete description of the START command.

The extended MCS console is established with the following attributes:

| Extended MCS Attribute | Value |
|---|---|
| Command Authority | System commands (SYS) |
| Console Key (used in DISPLAY C command) | STNBRNCH |
| Console Name | UNVnnnnn, where nnnnn is 00000 – 99999. |
| Command Scope | Current system |
| Message Scope | Current system |
| Override User Profile OPERPARM | Yes |

Extended MCS consoles can be protected so that only permitted users have the authority to issue commands. The RACF OPERCMDS class is used to establish user security for extended MCS consoles.

Refer to the IBM *MVS Planning: Operations* and the *Security Server RACF Security Administrator Guide* manuals for complete details.

## START System Command

The UCMD Manager provides the START command parameters. The STC CP adds parameter STDIN with a value of a cataloged dynamically allocated data set that contains the standard input from the Manager.

The syntax of the START command is as follows:

**S** ***manager-cmd*****,STDIN=***stdin-dataset***

The **manager-cmd** value is the command value provided by the UCMD Manager. The **stdin-dataset** value is the dynamically allocated data set that contains the Manager's standard input data.

As an example, the following Manager command, executed from a Windows system

```
ucmd –c "prdtask,opt=abc" –cmd_type stc –u ts0023 ...
```

results in a START command as follows:

```
S PRDTASK,OPT=ABC,STDIN=TS0023.UCM.C08AD835.STDIN
```

Access to UCMD Manager started task requests and the associated command value can be protected with Universal Access Control Lists. See Section 8.5.5 Universal Access Control List for complete details on protecting request types.

## Standard Input

A Manager can provide an input file to the started task via the UCMD Manager's standard input file. The Manager's standard input file is first spooled to a cataloged data set. The fully qualified data set name is passed to the started task as JCL procedure parameter STDIN.

The dynamically allocated stdin data set is allocated with a name formatted as follows:

***hlq*.UCM.C*cid*.STDIN**

where,

- **hlq** High-level qualifier is one of the following:
  - User ID with which the STC is executed.
  - Value of the configuration option STDIN_HLQ.
- **cid** Component ID of the STC CP. The value is the last seven of eight digits of the component ID in a hexadecimal format.

Standard input data sets dynamically allocated by the UCMD Server are deleted after the STC completes execution.

The UCMD Server's default stdin data set attributes are set with the DEFAULT_STDIN_ALLOC configuration option. The default values are DSORG=PS, RECFM=VB, LRECL=1024, UNIT=SYSDA, SPACE=(CYL,(5,5),RLSE). The UCMD Manager, optionally, can provide data set attributes using the Manager SERVER_OPTIONS value specifying the Server STDIN_ALLOC option described below.

Instead of the Manager providing a standard input file, the Manager may provide the name of an existing data set allocated on the Server system. That is accomplished with a Manager SERVER_OPTIONS value specifying the Server STDIN_ALLOC option described below.

## Standard Output and Error

The JES SYSOUT produced by the STC can be returned to the UCMD Manager as standard output and standard error. The STC JESLOG data (JESMSGLG, JESJCL, and JESYSMSG data) is returned as standard error. All step SYSOUT data is returned as standard output.

The STC CP will retrieve SYSOUT data after the STC completes execution. The SYSOUT must be spooled to the JES class specified by the UCMD Server JES_SELECT_CLAS option. Additionally, the SYSOUT data must be held. Released SYSOUT is not retrieved.

Each SYSOUT file is retrieve and written to the appropriate standard I/O file. Message UNV2435I prefixes each SYSOUT file. The message lists the ddname, step, procstep, and spool data set name of each SYSOUT file. The maximum number of records returned per SYSOUT file is controlled with the UCMD Server JES_MAX_LINES_READ configuration option.

After the SYSOUT files are retrieved, their disposition is controlled by the JES_DELETE_SPOOL_FILE and JES_REQUEUE_CLAS UCMD Server options.

## JCL Requirements

The started task JCL can specify a job or a procedure. Job JCL must come from either the IEFJOBS or IEFPDSI ddnames of the master JCL. Procedure JCL comes from either the IEFPDSI or JES procedure libraries.

In determining on whether to use job or procedure JCL, refer to the IBM *MVS JCL Reference* manual for a description of the advantages and disadvantages.

The first step of the started task must execute the Universal Started Task Support program, UCMSS000. The STC Support program establishes the user ID for the address space and performs necessary communication with the STC CP.

Figure 8.1, below, illustrates a started task procedure JCL.

```
//UCMREQ   PROC
//*
//UCMSS000 EXEC PGM=UCMSS000
//STEPLIB  DD   DISP=SHR,DSN=UNV.SUNVLOAD
//STDIN    DD   DISP=SHR,DSN=&STDIN
//SYSUDUMP DD   SYSOUT=H
//*
//S1       EXEC PGM=ABC123
//SYSOUT   DD  SYSOUT=A,HOLD=YES
//SYSPRINT DD  SYSOUT=A,HOLD=YES
//SYSIN    DD  DISP=SHR,DSN=NAH1A.JCL.CNTL(DATA)
```

Figure 8.1 Universal Command Server for z/OS - Started Task Procedure

The JCL executes two steps:

1. The first step executes the Universal Command Started Task Support program.
2. The second step executes program ABC123. (This second step - and any subsequent steps - can be any z/OS program.)

Note: The UCMSS00 step includes a STDIN ddname that uses the STDIN procedure variable. This is a JCL convention only to help eliminate one particular source of JCL errors when the source JCL is a procedure (not applicable for job JCL). Any procedure parameter (for example, STDIN) specified on the START command must be reference within the JCL. If it is not, a JCL error is the result. By using the STDIN JCL parameter in the first step, JCL errors caused by not using the parameter are eliminated. The UCMSS000 program does not attempt to use the STDIN ddname in any way.

The UCMSS000 program accepts one input parameter on the PARM keyword of the EXEC statement. The parameter SWUSR controls whether or not the address space user ID is switched or not. The format of the parameter is

**SWUSR={YES|NO}**

A value of **YES** specifies the user ID is switched. A value of **NO** specifies the user ID is not switched. The default is **YES**.

## 8.2.3 Command References

A command reference provides the ability to precisely define and control what is executed by the UCMD Server. The UCMD Manager does not provide the command or script. Everything is defined within the command reference. The command reference may optionally be defined to accept command or script options from the Manager.

Command references are defined as PDS members. The command reference PDS is allocated to the UNVCREF ddname of the Broker started task.

UCMD Managers refer to the command reference by member name and, optionally, provide an input file (via standard input) and options. A Manager requests the execution of a command reference by specifying a COMMAND_TYPE of **`cmdref`**. Command reference options are provided as they would for any command.

For example, the following UCMD Manager command can be used from Windows or UNIX to request execution of the command reference **`cref100`** and pass it options **`opt1`** and **`opt2`**:

```
ucmd –c “cref100 opt1,opt2” –cmd_type cmdref ...
```

z/OS command references can define any valid command type, such as USS shell commands and scripts and started task commands.

See Section 2.10 Command References for complete details on command references.

## USS Command Reference Example

The following command reference executes a **ucopy** command to read a file.

```
# Command reference to read a file.
#

-format cmd
-type   shell

<eof>

ucopy /opt/application/file.txt
```

## STC Command Reference Example

The following command reference starts started task SCHEDINT.

```
# Command reference to scheduler interface.
#

-format cmd
-type   stc

<eof>

SCHEDINT,OPT=ABC
```

# 8.3 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 UCMD Server for z/OS component definition is located in the component definition library **UNVCOMP** allocated to the Universal Broker ddname **UNVCOMP**. The UCMD Server component definition member is **UCMCMP00**.

Table 8.1, below, identifies all of the options that comprise the UCMD Server for z/OS component definition.

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

| Option Name | Description |
|---|---|
| AUTOMATICALLY_START | Specification for whether or not UCMD Server starts automatically when Universal Broker is started. |
| COMPONENT_NAME | Name by which the clients know the UCMD Server |
| CONFIGURATION_FILE | Member name of the UCMD Server configuration file in the **UNVCONF** library allocated to the Broker ddname **UNVCONF** |
| RUNNING_MAXIMUM | Maximum number of UCMD Servers that can run simultaneously |
| START_COMMAND | Member name of the UCMD Server program |
| WORKING_DIRECTORY | HMS directory used as the working directory of the UCMD Server |

Table 8.1 UCMD Server for z/OS - Component Definition Options

# 8.4 Configuration

Universal Command 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.

## 8.4.1 Manager Override

A UCMD Manager can specify certain UCMD Server configuration options when it makes its request for command execution to the UCMD Server. The UCMD Manager command line option **`-server`** (SERVER_OPTIONS) is used to specify UCMD Server options.

Which options are available for manager override depend on the UCMD Server platform and release. UCMD Server configuration options specify a manager override option only if applicable (see Chapter 3 Universal Command Server Configuration Options in the Universal Command 3.2.0 Reference Guide) . If override is not specified, no UCMD Manager override is available.

The UCMD Manager is not notified of override errors. The UCMD Server logs the error and continues processing the request.

## 8.4.2 Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation.

The UCMD Server configuration file name is specified in the Universal Command Server component definition. The default name is **`UCSCFG00`**. The name refers to a member in the PDS allocated to the Universal Broker ddname **`UNVCONF`**.

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

## 8.4.3 Configuration Options

Table 8.2, below, identifies all UCMD Server for z/OS configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Command 3.2.0 Reference Guide.

| Option Name | Description |
|---|---|
| ACTIVITY_MONITORING | Specification for whether or not product activity monitoring events are generated. |
| ALLOW_SPOOLING | Specification for whether or not spooling is permitted. |
| CODE_PAGE | Code page used for text translation. |
| COMMAND_TYPE | Default command type. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the control sessions. |
| DEFAULT_STDIN_ALLOC | Default STC standard input data allocation parameters. |
| EVENT_GENERATION | Events to be generated and processed as persistent events. |
| JES_DELETE_SPOOL_FILE | Specification for whether or not selected STC SYSOUT is deleted. |
| JES_MAX_LINES_READ | Maximum number of records read from selected STC SYSOUT. |
| JES_REQUEUE_CLAS | JES class in which selected STC SYSOUT is re-queued. |
| JES_SELECT_CLAS | JES class from which STC SYSOUT is selected. |
| JOB_RETENTION | Number of seconds that a disconnected server remains active after user process completes. |
| KEEPALIVE_INTERVAL | Specifies if and how often a keepalive message is sent. |
| LOGIN | Setup and login environment or not. |
| MESSAGE_LEVEL | Level of messages written. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not the server accepts the network fault tolerant protocol. |
| SCRIPT_TYPE | Script type of the user job being run. |
| SHELL | Default shell interpreter. |
| STDIN_ALLOC | STC standard input data set allocation parameters. |
| STDIN_HLQ | STC standard input data high-level qualifier. |
| STDIO_TIMEOUT | Wait time for standard I/O to close before the server process exits. |
| TMP_DIRECTORY | Name of the directory used for temporary files. |
| TRACE_FILE_LINES | Maximum number of lines to write to a trace file. |
| TRACE_TABLE | Memory trace table specification. |
| USE_USER_ACCOUNTING_CODE | Specification for whether or not user authentication is active. |

Table 8.2 UCMD Server for z/OS - Configuration Options

# 8.5 Security

Universal Command Server security concerns are:

1. Access to product data sets
2. Access to Universal Product configuration files
3. Universal Broker user account
4. Privacy and integrity of transmitted network data
5. User authentication

## 8.5.1 File Permissions

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

## 8.5.2 Configuration Files

Only trusted user accounts should have write permission to the Universal Command Server configuration files.

## 8.5.3 Universal Command Server User ID

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

## 8.5.4 User Authentication

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

# 8.5.5 Universal Access Control List

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

See Section 2.8 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 Command configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 8.3 identifies all Universal Command Server for z/OS UACL entries. Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Command 3.2.0 Reference Guide.

| UACL Entry Name | Description |
|---|---|
| UCMD_ACCESS | Allows or denies access to Universal Command Server services |
| UCMD_REQUEST | Allows or denies access to Universal Command Server services based on client identification and request type |

Table 8.3 Universal Command Server for z/OS - UACL Entries

# 8.5.6 UACL Entry Precedence

## Deny or Allow Access

The UCMD_ACCESS rules are searched first for an entry that matches the client request. If an UCMD_ACCESS entry is found and the rule denies access to the UCMD Manager, the search stops and the UCMD Manager request is denied.

If a UCMD_ACCESS entry is not found or a UCMD_ACCESS rule allows access, the UCMD_REQUEST entries are searched. If a UCMD_REQUEST entry is found, its rule determines whether the UCMD Manager request is denied or allowed.

If no rules are found, the UCMD Manager request is allowed.

## Authenticate or No Authenticate Access

The UCMD_ACCESS entries are searched followed by the UCMD_REQUEST entries.

If a UCMD_REQUEST entry is found, it sets the authentication requirement.

If a UCMD_REQUEST entry is not found and an UCMD_ACCESS entry is found, the UCMD_ACCESS rule sets the authentication requirement.

If no rules are found, the UCMD Manager request requires authentication.

Table 8.4, below, identifies the UACL entry precedence rules described above.

| ucmd_access | | ucmd_request | | Result |
|---|---|---|---|---|
| Allow/Deny | Auth/Noauth | Allow/Deny | Auth/Noauth | |
| NO-MATCH | NO-MATCH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| DENY | N/A | N/A | N/A | DENY |
| ALLOW | AUTH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | AUTH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | NOAUTH | ALLOW, NOAUTH |
| ALLOW | AUTH | DENY | N/A | DENY |
| ALLOW | NOAUTH | NO-MATCH | NO-MATCH | ALLOW, NOAUTH |
| ALLOW | NOAUTH | ALLOW | AUTH | ALLOW,AUTH |
| ALLOW | NOAUTH | ALLOW | NOAUTH | ALLOW,NOAUTH |

Table 8.4 Universal Command for z/OS - UACL Decision Table

## 8.5.7 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.

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

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

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user **TS1004** on that host. No host can execute commands as local user **SUPERID**. User **TS1004** on host 10.20.30.40 can execute commands as local user **TSUP1004** without providing the password. Users **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID **TSUP1004** without a password. Certificate ID **joe** is allowed to execute commands with any other local user ID with a password. Certificate ID operations cannot run anything. All other certificate IDs can execute commands with any user ID except for **SUPERID** with a password.

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

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

# Chapter 9
# Universal Command Server for Windows

## 9.1 Overview

This chapter documents the Universal Command (UCMD) Server at a detailed level. The material is specific to the Microsoft Windows operating system.

### 9.1.1 Server Environment

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

All components dependent upon Universal Broker (such as UCMD Server) inherit their message language from the Broker. All messages issued by components are sent to the Universal Broker for writing.

# 9.2 Commands

There are two types of work that a UCMD Server can execute:

1. Commands and Scripts
2. Command References

In all cases, the work executes in its own address space with its own user identity. No Universal Products programs share the address space with the unit of work started by the UCMD Server.

## 9.2.1 Command Environment

The user command executes in its own process. No Universal Products programs share the process with the user command. The process environment consists of several attributes that are described in this section.

### User Identification

UCMD Server can operate with user security active or inactive, as specified by the USE_USER_ACCOUNTING_CODE configuration option.

- With user security active, the UCMD Server requires the UCMD Manager to supply a valid local system user account or a domain account and a password. The user command executes with the user account identified by the supplied user ID.

  UCMD Managers specify a domain user account as **`DOMAIN\USER`**.
- With user security inactive, the UCMD Server does not require the UCMD Manager to supply a valid user ID. The user command executes with the user account of the UCMD Server. The user account of the UCMD Server is inherited from the Universal Broker.

  The Universal Broker service runs with the Local System account. The Local System account provides a lot of access to the operating system that a user process typically does not require. Setting security inactive is not recommended because of the level of access it permits the user process.

## Working Directory

UCMD Server can operate with user security active or inactive, as specified by the USE_USER_ACCOUNTING_CODE configuration option.

- With user security active, a user command's working directory is a subdirectory of the Universal Command Home directory, which defaults to **`\Program Files\Universal\UcmdHome`**. The name of the subdirectory is the user ID with which the command executes. For example, if user HOGIN executes a command via Universal Command, the commands working directory is **`\Program Files\Universal\UcmdHome\HOGIN`**.

  If the working directory is not defined when the user command executes, the UCMD Server creates the directory before it executes the user command.
- With user security inactive, a user command's working directory is the UCMD Server's working directory. All user commands executed use the same directory. Care should be taken to avoid name clashes or other consequences of multiple processes sharing a working directory.

## Command Shell

The default command interpreter used to execute commands and scripts is **`CMD.EXE`**. This commonly is referred to as the DOS command processor. The path to the **`CMD.EXE`** program is obtained from the **`COMPSPEC`** environment variable, or if **`COMPSPEC`** is not defined, the path is derived from the **`WINDIR`** environment variable as **`%WINDIR\system32\cmd.exe`**. If that fails, the server exits with an error.

Manager-supplied script files are processed as batch files (extension .BAT) by default. The file type (that is, the extension) can be changed with the SCRIPT_TYPE UCMD Server configuration option or the SCRIPT_TYPE UCMD Manager option.

## Environment Variables

UCMD Server inherits its environment variables from the Universal Broker. In turn, the user command inherits its environment variables from the UCMD Server. The UCMD Server does not add, delete, or edit any environment variables.

## 9.2.2 Command References

A command reference provides the ability to precisely define and control what is executed by the UCMD Server. The UCMD Manager does not provide the command or script. Everything is defined within the command reference. The command reference may optionally be defined to accept command or script options from the UCMD Manager.

Command references are defined as files in the command reference directory as defined by the CMD_REFERENCE_DIRECTORY UCMD Server option.

UCMD Managers refer to the command reference by file name and, optionally, provide an input file (via standard input) and options. A UCMD Manager requests the execution of a command reference by specifying a COMMAND_TYPE of *cmdref*. Command reference options are provided as they would be for any command.

For example, the following UCMD Manager command can be used from Windows or UNIX to request execution of the command reference **`cref100`** and pass it options **`opt1`** and **`opt2`**:

```
ucmd -c "cref100 opt1,opt2" -cmd_type cmdref ...
```

Command references can define any valid command type, such as commands and scripts.

See Section 9.2.2 Command References for complete details on command references.

### Command Reference Example

The following command reference executes a **`ucopy`** command to read a file.

```
# Command reference to read a file.
#

-format cmd
-type   shell

<eof>

ucopy \application\file.txt
```

# 9.3 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 9.1, below).

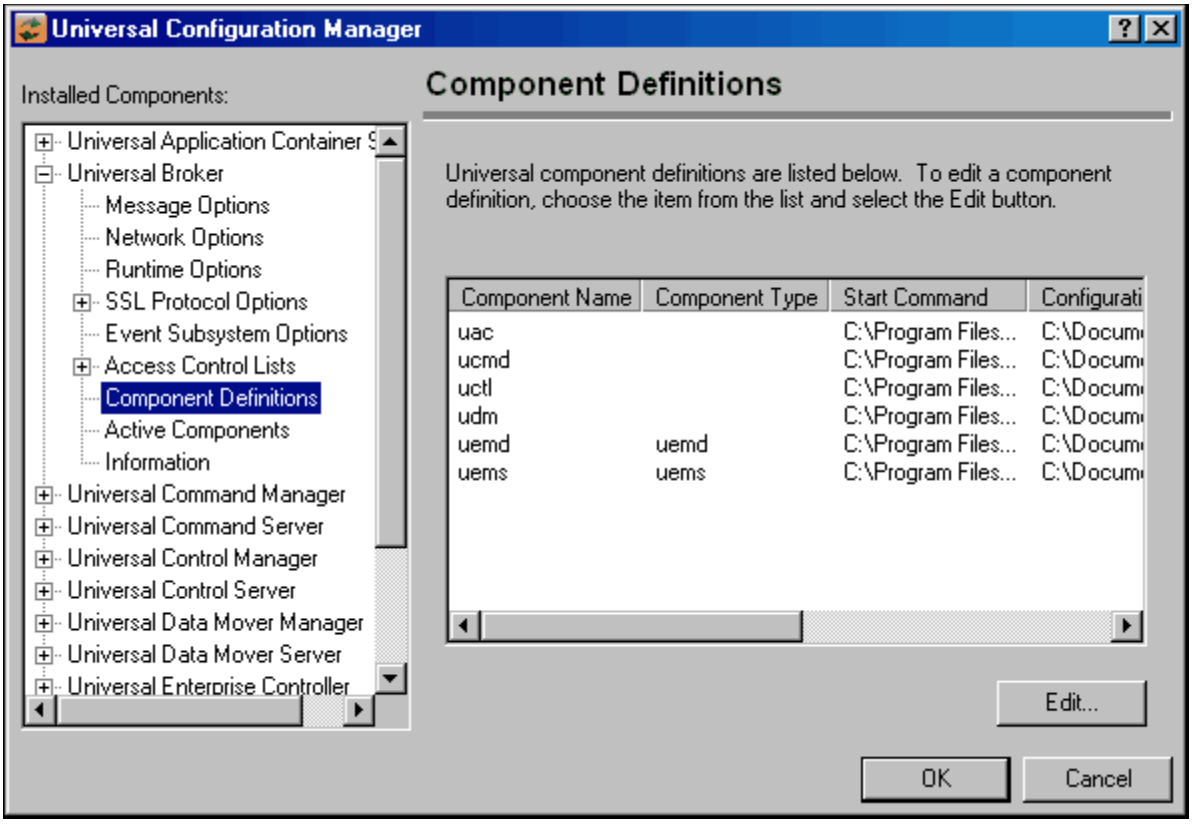


Figure 9.1 Universal Configuration Manager - Component Definitions

Table 9.1, below, identifies all of the options that comprise the UCMD Server for Windows component definition.

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

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

Table 9.1 UCMD Server for Windows - Component Definition Options

# 9.4 Configuration

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

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

## 9.4.1 Manager Override

A UCMD Manager can specify certain UCMD Server configuration options when it makes its request for command execution to the UCMD Server. The UCMD Manager command line option **`-server`** (SERVER_OPTIONS) is used to specify UCMD Server options.

Which options are available for manager override depend on the UCMD Server platform and release. UCMD Server configuration options specify a manager override option (see Chapter 3 Universal Command Server Configuration Options) only if applicable. If override is not specified, no UCMD Manager override is available.

The UCMD Manager is not notified of override errors. The UCMD Server logs the error and continues processing the request.

## 9.4.2 Configuration File

The configuration file provides a simple method of specifying configuration values that will not change with each command invocation.

The UCMD Server configuration file name is specified in the UCMD Server component definition. The default name is **`ucmds.conf`**.

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).

Note: For any changes made directly to the UCMD 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.

## 9.4.3 Configuration Options

Table 9.2, below, identifies all UCMD Server for Windows configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Command 3.2.0 Reference Guide.

| Option Name | Description |
|---|---|
| ACTIVITY_MONITORING | Specification for whether or not product activity monitoring events are generated. |
| ALLOW_SPOOLING | Specification for whether or not spooling is permitted. |
| ASSIGN_PROCESS_TO_JOB | Specification for whether or not UCMD Server assigns child processes to a single Windows job object. |
| CMD_REFERENCE_DIRECTORY | Command reference directory. |
| CODE_PAGE | Code page used for text translation. |
| COMMAND_TYPE | Default command type. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the control sessions. |
| EVENT_GENERATION | Events to be generated and processed as persistent events. |
| INSTALLATION_DIRECTORY | Base directory in which UCMD Server is installed. |
| INTERACT_WITH_DESKTOP | Specification for whether or not the desktop of the current interactive logon session is accessible to the user process. |
| JOB_RETENTION | Number of seconds a disconnected server remains active after user process completes. |
| KEEPALIVE_INTERVAL | Frequency with which a keepalive message is sent. |
| LOGIN | Specification for whether to load the user's profile and environment. |
| 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. |
| SCRIPT_TYPE | Default script type. |
| SPOOL_DIRECTORY | Spool file directory. |
| STDIO_TIMEOUT | Wai time for standard I/O to close before the server process exits. |
| TMP_DIRECTORY | Name of the directory used for temporary files. |
| TRACE_DIRECTORY | Trace file directory. |
| TRACE_FILE_LINES | Maximum number of lines to write to a trace file. |
| TRACE_TABLE | Memory trace table specification. |
| USE_USER_ACCOUNTING_CODE | Specification for whether or not user authentication is active. |

Table 9.2 UCMD Server for Windows - Configuration Options

# 9.5 Security

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

Universal Command security concerns are:

1. Access to Universal Command files and directories
2. Access to Universal Command configuration options
3. Universal Command Server user account
4. Privacy and integrity of transmitted network data
5. User authentication

## 9.5.1 File Permissions

Only trusted user accounts should have write permission to the UCMD Server installation directory and subdirectories, and all of the files within them. This most likely means only the administrator group should have write access. Eligible users of Universal Command require read access to the message catalogs (`*.umc` files) in the `nls` subdirectory of the Universal Products installation directory.

All eligible users of Universal Command require permission to create directories in the UCMD Server working directory, if security is activated. A directory named after the user ID requesting the command is created for each user. The directory is created while impersonating the user; hence, it's created using the user's security account.

Home directories are created with permissions giving the user full control of both the directory and the files within the directory.

## 9.5.2 Configuration Files

Only trusted user accounts should have write permission to the UCMD Server configuration files, and add and delete access to the directories in which they reside.

Although configuration files can be edited with any text editor (for example, Notepad), the Universal Configuration Manager application is the recommended way to set configuration options. The Universal Configuration Manager can be executed only by accounts in the Administrator group (see Section 2.4 Universal Configuration Manager).

## 9.5.3 User Authentication

User authentication is the process of verifying that a user is a known and valid user. The process used by Universal Command Server requires the user to provide a WIndows user name / ID and a password. The Universal Command 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 for them to do a batch log on. All users can do an interactive log on. See the LOGON_METHOD option for more details.

## 9.5.4 Universal Access Control List

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

See Section 2.8 Universal Access Control List for details on the Universal Access Command List feature.

### UACL Entries

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

Table 9.3 identifies all Universal Command for Windows UACL entries.

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

| UACL Entry Name | Description |
|---|---|
| UCMD_ACCESS | Allows or denies access to Universal Command Server services |
| UCMD_REQUEST | Allows or denies access to Universal Command Server services based on client identification and request type |

Table 9.3 Universal Command for Windows - UACL Entries

## Updating the Universal Command 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 9.2, 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.

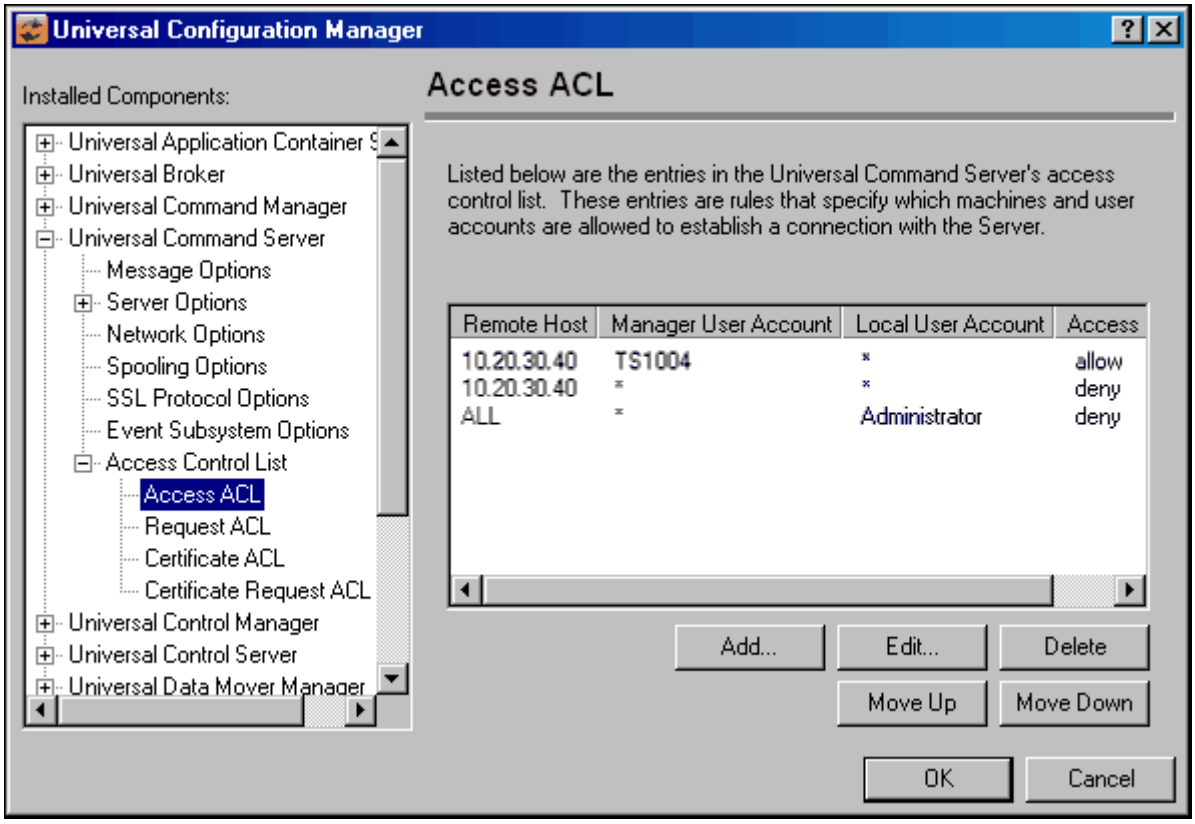


Figure 9.2 Universal Configuration Manager - Universal Command Server - Access ACL

# Chapter 10
# Universal Command Server for UNIX

## 10.1 Overview

This chapter documents the Universal Command (UCMD) Server at a detailed level. The material is specific to the UNIX variety of operating systems.

### 10.1.1 Server Environment

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

All components dependent upon Universal Broker (such as UCMD Server) inherit the message language from the Universal Broker. All messages issued by components are sent to the Universal Broker for writing.

# 10.2 Commands

There are two types of work that a UCMD Server can execute:

1. Commands and Scripts
2. Command References

In all cases, the work executes in its own address space with its own user identity. No Universal Products programs share the address space with the unit of work started by the UCMD Server.

## 10.2.1 User Identification

UCMD Server can operate with user security active or inactive, as specified by the USER_SECURITY configuration option.

- With user security active, the UCMD Server requires the UCMD Manager to supply a valid user ID for the local system and a password. The user command executes with the user ID and the primary and secondary group ID's of the user.
- With user security inactive, the UCMD Server does not require the UCMD Manager to supply a valid user ID. The user command executes with the user account of the UCMD Server. The user account of the UCMD Server is the superuser account (UID 0).

The superuser account provides a lot of access to the operating system that a user process typically does not require. Setting security inactive is not recommended because of the level of access it permits the user process.

## 10.2.2 Working Directory

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

- With user security active, a user command's working directory is the home directory of the user ID as defined in the **`/etc/passwd`** file.
- With user security inactive, a user command's working directory is the UCMD Server's working directory. All user commands executed use the same directory. Care should be taken to avoid name clashes and other consequences of multiple processes sharing a working directory.

## 10.2.3 Command Shell

The UCMD Manager LOGIN option and UCMD Server LOGIN option specify what shell is used.

For non-login environments, the default is shell **/bin/sh**. The shell used for non-login environments is configurable with the SHELL option.

For login environments, the shell associated with the user ID found in the **/etc/passwd** file is used. The shell environment is created as if the user logged on interactively. For example, the **korn** shell's **.profile** is used to initialize the environment.

The non-login environment is similar to the environment the **cron** scheduler provides. User resource files, such as **.profile**, are not utilized.

The application scripts being executed and your local system management policies should be used to determine which method is best.

If user security is inactive, the default shell **/bin/sh** always is used independent of the LOGIN option.

## 10.2.4 Environment Variables

Environment variables are inherited from the server, which in turn inherits them from the broker.

If security is active, certain variables are modified to match the user environment. They are HOME, LOGNAME, USER, PWD, and SHELL. Their values are updated to reflect the values for the new environment.

The following variables are added if not found in the environment: HOME, USER, SHELL, and UCMDENV. The UCMDENV variable is set to a value of *1*. It can be used within scripts to determine if Universal Command has invoked them.

The UCMD Manager LOGIN option and UCMD Server LOGIN option have an impact on the environment variables defined. For login environments, the user's shell is invoked as a login shell, which, in turn, uses the shell **.profile** file in the user's home directory. Therefore, any environment variables set in the **.profile** file also will be defined. UCMD Server inherits its environment variables from the Universal Broker. In turn, the user command inherits its environment variables from the UCMD Server.

If user security is inactive, no changes are made to the environment variables.

## 10.2.5 Command References

A command reference provides the ability to precisely define and control what is executed by the UCMD Server. The UCMD Manager does not provide the command or script. Everything is defined within the command reference. The command reference may optionally be defined to accept command or script options from the UCMD Manager.

Command references are defined as files in the command reference directory as defined by the CMD_REFERENCE_DIRECTORY UCMD Server option.

UCMD Managers refer to the command reference by file name and, optionally, provide an input file (via standard input) and options. A UCMD Manager requests the execution of a command reference by specifying a COMMAND_TYPE of *cmdref*. Command reference options are provided as they would be for any command.

For example, the following UCMD Manager command can be used from Windows or UNIX to request execution of the command reference **`cref100`** and pass it options **`opt1`** and **`opt2`**:

```
ucmd -c "cref100 opt1,opt2" -cmd_type cmdref ...
```

Command references can define any valid command type, such as commands and scripts.

See Section 2.10 Command References for complete details on command references.

### Command Reference Example

The following command reference executes a **`ucopy`** command to read a file.

```
# Command reference to read a file.
#
-format cmd
-type   shell

<eof>
ucopy /application/file.txt
```

# 10.3 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 UCMD Server for UNIX component definition is located in the component definition directory of the Universal Broker.

Table 10.1, below, identifies all of the options that comprise the UCMD Server for UNIX component definition.

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

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

Table 10.1 UCMD Server for UNIX - Component Definition Options

# 10.4 Configuration

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

See Section 2.1 Configuration for details on Universal Products configuration methods.

## 10.4.1 Manager Override

A UCMD Manager can specify certain UCMD Server configuration options when it makes its request for command execution to the UCMD Server. The UCMD Manager command line option **`-server`** (SERVER_OPTIONS) is used to specify UCMD Server options.

Which options are available for manager override depend on the UCMD Server platform and release. UCMD Server configuration options specify a manager override option (see Chapter 3 Universal Command Server Configuration Options) only if applicable. If override is not specified, no UCMD Manager override is available.

The UCMD Manager is not notified of override errors. The UCMD Server logs the error and continues processing the request.

## 10.4.2 Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation.

The UCMD Server configuration file name is specified in the Universal Command Server component definition. The default name is **`ucmds.conf`**. See the component definition file to determine the directory in which it is located. This file can be edited manually with any text editor.

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

## 10.4.3 Configuration Options Summary

Table 10.2, below, identifies all UCMD Server for UNIX configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Command 3.2.0 Reference Guide.

| Option Name | Description |
|---|---|
| ACTIVITY_MONITORING | Specification for whether or not product activity monitoring events are generated. |
| ALLOW_SPOOLING | Specifies whether or not spooling in permitted. |
| CMD_REFERENCE_DIRECTORY | Command reference directory. |
| CODE_PAGE | Code page used for text translation. |
| COMMAND_TYPE | Default command type. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the control sessions. |
| EVENT_GENERATION | Events to be generated and processed as persistent events. |
| INSTALLATION_DIRECTORY | Base directory in which UCMD Server is installed. |
| JOB_RETENTION | Number of seconds a disconnected server remains active after user process completes. |
| KEEPALIVE_INTERVAL | Specifies if and how often a keepalive message is sent. |
| LOGIN | Setup and login environment or not. |
| MESSAGE_LEVEL | Level of messages written. |
| NETWORK_FAULT_TOLERANT | Specifies whether or not the server accepts the network fault tolerant protocol. |
| NLS_ DIRECTORY | Location of UMC and UTT files. |
| SCRIPT_TYPE | Script type of the user job being run. |
| SHELL | Specifies the default shell interpreter. |
| SPOOL_DIRECTORY | Location of spool files. |
| STDIO_TIMEOUT | Specifies the time in seconds to wait for Standard I/O to close before the server process exits. |
| TMP_DIRECTORY | Name of the directory used for temporary files. |
| TRACE_DIRECTORY | Location of trace files. |
| TRACE_FILE_LINES | Maximum number of lines to write to a trace file. |
| TRACE_TABLE | Memory trace table specification. |
| USE_USER_ACCOUNTING_CODE | Specifies if user authentication is active or not. |

Table 10.2 UCMD Server for UNIX - Configuration Options

# 10.5 Security

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

Universal Command security concerns are:

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

## 10.5.1 File Permissions

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

## 10.5.2 Configuration Files

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

## 10.5.3 Universal Command Server User ID

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

See the Universal Message Translator chapter in the Universal Products Utilities 3.2.0 User Guide for details.

## 10.5.4 User Authentication

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

Universal Command can use three different types of user authentication methods:

1. Default authentication uses the UNIX traditional password comparison method.
2. PAM authentication uses the PAM API to authenticate users. This option is only available for certain UNIX platforms.
3. HP-UX Trusted Security uses HP-UX Trust Security APIs to authenticate users. This is only available on Hewlett Packard HP-UX and Tru64 platforms.

## 10.5.5 Universal Access Control List

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

See Section 2.8 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 Command configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 10.3 identifies all Universal Command for UNIX UACL entries. Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Command 3.2.0 Reference Guide.

| UACL Entry Name | Description |
|---|---|
| UCMD_ACCESS | Allows or denies access to Universal Command Server services |
| UCMD_REQUEST | Allows or denies access to Universal Command Server services based on client identification and request type |

Table 10.3 Universal Command Server for UNIX - UACL Entries

# UACL Entry Precedence

## Deny or Allow Access

The **ucmd_access** rules are searched first for an entry that matches the client request. If an **ucmd_access** entry is found and the rule denies access to the Manager, the search stops and the Manager request is denied.

If no **ucmd_access** entry is found or an **ucmd_access** rule allows access, the **ucmd_request** entries are searched. If an **ucmd_request** entry is found, its rule determines whether the Manager request is denied or allowed.

If no rules are found, the Manager request is allowed.

## Authenticate or No Authenticate Access

The **ucmd_access** entries are searched followed by the **ucmd_request** entries.

If an **ucmd_request** entry is found, it sets the authentication requirement.

If no **ucmd_request** entry is found and an **ucmd_access** entry is found, the **ucmd_access** rule sets the authentication requirement.

If no rules are found, the Manager request requires authentication.

Table 10.4, below, identifies the UACL entry precedence rules described above.

| ucmd_access | | ucmd_request | | Result |
|---|---|---|---|---|
| **Allow/Deny** | **Auth/Noauth** | **Allow/Deny** | **Auth/Noauth** | |
| NO-MATCH | NO-MATCH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| DENY | N/A | N/A | N/A | DENY |
| ALLOW | AUTH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | AUTH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | NOAUTH | ALLOW, NOAUTH |
| ALLOW | AUTH | DENY | N/A | DENY |
| ALLOW | NOAUTH | NO-MATCH | NO-MATCH | ALLOW, NOAUTH |
| ALLOW | NOAUTH | ALLOW | AUTH | ALLOW,AUTH |
| ALLOW | NOAUTH | ALLOW | NOAUTH | ALLOW,NOAUTH |

Table 10.4 Universal Command Server for UNIX - UACL Decision Table

## 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.

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

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

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host but has limited access from host 10.20.30.40 to user TS1004 on that host. No host can execute commands as local user root. User TS1004 on host 10.20.30.40 can execute commands as local user tsup1004 without providing the password. Users TS1004 from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID tsup1004 without a password. Certificate ID **`joe`** is allowed to execute commands with any other local user ID with a password. Certificate ID **`operations`** cannot run anything. All other certificate IDs can execute commands with any user ID except for root with a password.

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

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

# Chapter 11
# Universal Command Server for OS/400

## 11.1 Overview

This section documents the Universal Command Server at a detailed level. The material is specific to the OS/400 operating systems.

### 11.1.1 Server Environment

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

All components dependent upon Universal Broker (such as Universal Command Server) inherit the message language from the Broker. All messages issued by components are sent to the Universal Broker for writing.

# 11.2 Commands

## 11.2.1 User Command Environment

The user request is initiated by the Universal Command Server Initiator program (**UCMSINIT**) running under the **UCMSINIT** pre-start job via the **UCMSRV** program running under the Universal Command Server (**UCMSRV**) job.

The UCMSINIT program:

1. Performs environment setup required to execute user commands or scripts.
2. Redirects the job log as requested.
3. Changes the user profile if user security is active.
4. Initiates the user request.
5. Monitors the user's request as it runs.
6. Catches any exceptions that occur.

Following completion of the users request, the **UCMSRV** program:

1. Processes the **UCMSINIT** job logs as required.
2. Returns the **UCMSINIT** job logs to the Universal Command Manager as requested.
3. Cleans up the environment.

The **UCMSRV** program also handles fault tolerant requests from the Universal Command Manager.

The **UCMSINIT** job log is returned to the Universal Command Manager and written to standard error. The spool file output produced by the executed commands is written to the user's default spool output queue. The spool files produced by the commands are not returned to the Universal Command Manager. They are left in the spool. If you would like the command output returned to the Universal Command Manager as well as the job log, See the Universal Submit Job chapter in the Universal Products Utilities 3.2.0 User Guide for an execution method that returns all command spool files as well as the job log.

Both the **UCMSINIT** and **UCMSRV** pre-start jobs are defined in the **UNVUBR320** subsystem. By default, there are always at least three each of the **UCMSINIT** and UCMSRV pre-start jobs running under the **UNVUBR320** subsystem.

# Initiator (UCMSINIT) Exit Points

The Initiator (**UCMSINIT**) calls two user exits:

1. **UCMSJOBI** is called once for job initialization.
2. **UCMSJOBT** is called once for job termination.

The CL source code is provided in **UNVPRD320/UNVCLSRC**.

The CL source files are compiled and bound with the following command:

```
CRTBNDCL PGM(UNVPRD320/exitname)
              SRCFILE(UNVPRD320/UNVCLSRC)
              SRCMBR(exitname)
```

Change the **exitname** to the name of the exit to be compiled and bound.

## UCMSJOBI

The **UCMSJOBI** exit is called before any user command is executed. This exit can be used to customize the job's environment to meet local requirements. It executes under the user profile requested by the Manager. If the exit issues unhandled messages with a severity greater than or equal to the value of the END_SEVERITY option of the UCMD Server, the job will terminate without executing any user commands.

**UCMSJOBI** sets the current library to the current library specified in the user profile under which the job runs. However, if the UCMD Server for OS/400 LOGIN option is enabled, there may be a conflict between the current library set by **UCMSJOBI** and by the LOGIN functionality. To avoid this conflict, a new **UCMSJOBI LOGIN** parameter specifies that when LOGIN is active, **UCMSJOBI** no longer sets the current library.

If the 3.2.0 version of **UCMSJOBI** is used to replace **UCMSJOBI** on a 3.1.1 system, an exception will occur because of the new **LOGIN** parameter.

Note: If **UCMSJOBI** from a previous release (3.1.1) of UCMD Server for OS/400 is copied in place of the 3.2.0 version, the program will run with the potential conflict.

## UCMSJOBT

The **UCMSJOBT** exit is called after all user commands have completed. The current exit code is passed in as a parameter. The exit executes under the user profile requested by the UCMD Manager. The exit will always be called once **UCMSJOBI** returns successfully. If **UCMSJOBI** issues an unhandled message that causes job termination, **UCMSJOBT** will not be called.

## User Command Exit Code

The exit code returned to the UCMD Manager indicates the success or failure of the user-requested command. The exit code is returned to the UCMD Manager by the UCMD Server (UCMSRV) based on the exit code that it receives from the Initiator (**UCMSINIT**). The Initiator sets its exit code based on the highest severity of the OS/400 messages propagated to it from user commands or scripts. The Initiator traps and handles all *ESCAPE, *NOTIFY, *STATUS messages and function checks.

**UCMSINIT** continues executing user commands as long as the highest message severity is less than the severity specified by the END_SEVERITY option of the UCMD Server.

In the event of an error not associated with the user-requested command, **UCMSINIT** returns exit code 99. If the error occurs following set up for returning the **UCMSINIT** job log to the user, the job log is returned as usual. Otherwise, no job log is returned and the user must check the output queues for a job log associated with the failure.

Depending on the job's logging settings and the severity of the error, no job jog may be saved. If an exception results in job termination, the returned exit code will be based on the numeric portion of the message identifier. The last four hexadecimal digits are used for the return code, with the most common being 9901 (decimal 39169). However, partial truncation, to the lower three digits, also occurs for managers running on some platforms.

If a job ends as a result of the **ENDJOB** command, whether issued directly or indirectly, the exit code will be the special value 199.

### User Identification

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

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

## 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 and working directory is the home directory of the user profile specified in the UCMD Manager.
- With user security inactive, the current library and the working directory are those for the user profile associated with the service program. The default user profile defined and associated with the service program at installation is **UNVUBR320**.

Note: The default value used for the current library in the **UNVUBR320** user profile is **UNVTMP320**. Care should be taken to avoid name clashes and other consequences of multiple processes sharing a common current library and working directory.

## 11.2.2 User Commands

UCMD Server accepts four forms of commands from a UCMD Manager.

1. Single CL command
2. Single REXX line
3. CL command file
4. REXX EXEC file

### Single CL Command

The remote UCMD Manager specifies a CL command using the COMMAND (**-cmd**) option.

The CL command must be of a type that can be executed by the QCMDEXC API. This is indicated by the command description in the CL Reference manual by the keyword Exec in the upper right corner of the command's syntax diagram. Limit **-cmd** option length to 1000 bytes.

### Single REXX Line

The remote UCMD Manager specifies a single REXX line using the COMMAND (**-cmd**) option with the COMMAND_TYPE (**-cmd_type**) option of *rexx*.

REXX and any associated commands must be of a type that can be executed by the QCMDEXC API as described in Single CL Command above. Limit COMMAND option length to 1000 bytes.

Multiple statements contained in the single line command must be separated by semicolons as described in the REXX manuals. The first statement does no require a REXX comment.

For example, the following command sends the text "'Change current library to ABC" to standard output and changes the current library:

```
ucmd -c "say 'Change current library to ABC'; \"CHGCURLIB
CURLIB(ABC) \"" -i as400 -u qsysopr -w qsysopr -cmd_type rexx
```

A user may use a simple REXX program in this context to setup and execute programs on the OS/400.

REXX provides the benefit of using standard output (STDOUT) and standard input (STDIN) files as part of their environment. The SAY command writes to STDOUT and the PULL command reads from STDIN.

STDOUT from REXX on the OS/400 is redirected back to STDIN of the Command Manager and REXX STDIN on the OS/400 is redirected from the STDOUT of the Command Manager. CL command files do not use STDOUT or STDIN directly.

## CL Command File

The remote UCMD Manager specifies a CL command file using the SCRIPT_FILE (-script) option.

The command file contains a sequence of CL commands to be executed in sequential order. The commands are executed from first to last or until a command generates a message with a severity greater than or equal to the END_SEVERITY option of the UCMD Server.

The CL commands are limited to the same set of CL commands described in Single CL commands above.

Each command is executed within the same job environment. This is similar to a batch job execution, but // CL commands cannot be used.

Blank lines and CL comment lines are ignored in the command file.

CL line continuation characters (+ and -) can be used in the command file and are processed accordingly.

The first line cannot be a comment line containing the word REXX.

## REXX EXEC File

The remote UCMD Manager specifies a REXX EXEC file by using the SCRIPT_FILE (-script) option.

This same UCMD Manager option is used to specify a CL command file. To distinguish between a REXX EXEC and a CL command file, the first line of the file containing the REXX EXEC must be a comment line containing the word REXX. The case of the letters does not matter.

For example, the following line is sufficient to indicate a REXX EXEC file:

```
/* REXX */
```

REXX EXECs have the benefit of using standard output (STDOUT) and standard input (STDIN) files as part of their environment. The SAY command writes to STDOUT and the PULL command reads from STDIN.

The STDOUT file is redirected back to the STDOUT of the UCMD Manager and the STDIN is redirected from the STDIN of the UCMD Manager. CL command files do not directly use STDOUT and STDIN.

# 11.3 Command References

A command reference provides the ability to precisely define and control what is executed by the UCMD Server. The UCMD Manager does not provide the command or script. Everything is defined within the command reference. The command reference may optionally be defined to accept command or script options from the UCMD Manager.

Command references that execute OS/400 commands or REXX scripts are defined as files in the UNVCMDREF library. Each file can contain only one member. For security reasons, the library name is set to UNVCMDREF; this name cannot be changed or redefined.

Managers refer to the command reference by file name and optionally provide an input file (via standard input) and options.

A Manager requests the execution of a command reference by specifying a COMMAND_TYPE of *cmdref*. Command reference options are provided as they would for any command.

For example, the following UCMD Manager command can be used from Windows or UNIX to request execution of the command reference **`cref100`** and pass it options **`opt1`** and **`opt2`**:

```
ucmd -c "cref100 opt1,opt2" -cmd_type cmdref ...
```

OS/400 command references can define command types **`cmd`** and **`rexx`**.

See Section 2.10 Command References for complete details on command references.

## 11.3.1 Command Reference Example

The following command reference contains a command to display a library catalog.

To use this **cmdref**, invoke UCMD Manager using:

**ucmd -c "cmdref_cmd" -cmd_type cmdref -u xxxx -w zzzz -i as400**

In this case, the user (**xxxx**) has authority to call the OS/400 system object QWCRJBST and **cmdref_cmd** is the name of the command reference file on the OS/400.

```
# -- File named cmdref_cmd in library UNVCMDREF --
# Use USBMJOB to execute the DSPLIB command.
#
-format cmd
-type cmd
<eof>
usbmjob cmd(dsplib qsysopr)
```

The next command reference contains a series of four commands which are executed in sequence from top to bottom.

Invoke these commands from system as4test using:

ucmd -c "cref_test" -cmd_type cmdref -u xxxx -w zzzz -i as4test

In this case, the user (**xxxx**) has authority to call the OS/400 system object QWCRJBST and **cref_test** is the name of the command reference file on the OS/400.

```
# -- File named cref_test in library UCVCMDREF --
# Execute a series of commands.  The output of the first two will remain
# on the output queue associated with the job's printer.  The output of
# the second two (by means of USBMJOB) will be directed to standard output
# and sent to the system running the Universal Command Manager.
#
-format script
-type cmd
<eof>

DSPUSRPRF USRPRF(QUSER)
dspcurdir
usbmjob cmd(dsplib quser)
usbmjob cmd(dsplibl)
```

## 11.3.2 REXX Command Reference Example

The following command reference sends the message and the library catalog to standard output. The job logs are sent to standard error.

To invoke this **cmdref** on system denver, use:

```
ucmd -c "rexx_test" -cmd_type cmdref -u xxxx -w zzzz -i denver
```

In this case, the user (**xxxx**) has authority to call the OS/400 system object QWCRJBST and **rexx_test** is the name of the command reference file on the OS/400.

Once again, the user, designated by the -u option, requires access to the object QWCRJBST. Without this access, **usbmjob** will fail.

```
# -- File named rexx_test in library UCVCMDREF --
# Use USBMJOB to execute the DSPLIB command.
#
-format script
-type rexx
<eof>
say "Submitting job to display library qsysopr"
'usbmjob cmd(dsplib qsysopr)'
```

# 11.4 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.

UCMD Server for OS/400 component definition is located in the component definition file of the Universal Broker. The default location for Universal Broker component definition files is **UNVPRD320/UNVCOMP**. The UCMD Server component member is **UCMD**.

Table 11.1, below, identifies all of the options that comprise the UCMD for OS/400 component definition.

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

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

Table 11.1 UCMD Server for OS/400 - Component Definition Options

# 11.5 Configuration

UCMD 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.

## 11.5.1 Manager Override

A UCMD Manager can specify certain UCMD Server configuration options when it makes its request for command execution to the UCMD Server. The UCMD Manager command line option -server is used to specify UCMD Server options.

Which options are available for UCMD Manager override depend on the UCMD Server platform and release. The configuration options listed below describe the UCMD Manager override option only if applicable. If the option is not listed, than no UCMD Manager override is available.

The UCMD Manager is not notified of override errors. The UCMD Server logs the error and continues processing the request.

## 11.5.2 Configuration File

The configuration options provide the simplest method of specifying configuration values that will not change with each command invocation.

Configuration options are specified in the UCMD Server configuration file. The configuration file name is specified in the UCMD Server component definition. The default file name is **`UNVPRD320/UNVCONF(UCMDS)`**. This file can be edited manually with any text editor (for example, Notepad or Source Edit Utility (SEU)).

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

## 11.5.3 Configuration Options Summary

Table 11.2, below, identifies all UCMD Server for OS/400 configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Command 3.2.0 Reference Guide.

| Option Name | Description |
|---|---|
| ACTIVITY_MONITORING | Specification for whether or not product activity monitoring events are generated. |
| ALLOW_SPOOLING | Specification for whether or not spooling is permitted. |
| CODE_PAGE | Code page used for text translation. |
| COMMAND_TYPE | Default command type. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| DATA_SSL_CIPHER_LIST | SSL cipher list for the control sessions. |
| END_SEVERITY | Message severity that terminates the initiator. |
| EVENT_GENERATION | Events to be generated and processed as persistent events. |
| JOB_LOG | Job log processing. |
| JOB_RETENTION | Number of seconds that a disconnected server remains active after user process completes. |
| KEEPALIVE_INTERVAL | Frequency of how often a keepalive message is sent. |
| LOGIN | Specification for whether or not to set up a login environment. |
| MESSAGE_LEVEL | Level of messages written. |
| NETWORK_FAULT_TOLERANT | Specification for whether or not the server accepts the network fault tolerant protocol. |
| SCRIPT_TYPE | Script type of the user job being run. |
| STDIO_TIMEOUT | Length of time to wait for standard I/O to close before the server process exits. |
| TRACE_FILE_LINES | Maximum number of lines to write to a trace file. |
| TRACE_TABLE | Memory trace table specification. |
| USE_USER_ACCOUNTING_CODE | Specification for whether or not the OS/400 user profile under which a process is run is to be used as the source for the job accounting code |
| USE_USER_ACCOUNTING_CODE | Specification for whether or not user authentication is active. |

Table 11.2 UCMD Server for OS/400 - Configuration Options

# 11.6 Security

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

Universal Command security concerns are:

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

## 11.6.1 Object Permissions

Only administrator accounts should have write permission to the Universal Command Server product library, **UNVPRD320**; the product temporary library, **UNVTMP320**; the command reference library, **UNVCMDREF**; the universal spool library, **UNVSPL320** and all objects within these libraries. For maximum security, only trusted accounts (administrators and the **UNVUBR320** profile) should have management, existence, alter, add, update or delete authority to these objects. As a reminder, the system value **QCRTAUT** controls public access authority to created objects unless overridden by specific commands.

## 11.6.2 Universal Command Server User Profile

If user security is activated, the Server requires, by default, *ALLOBJ authority to change user profiles. Unless modifications are made as described in Section Removing *ALLOBJ Authority from **UNVUBR320** User Profile, the Server user profile, which is inherited from the Broker, requires *ALLOBJ authority.

## 11.6.3 User Authentication

User authentication is the process of verifying that a user is a known and valid user. The process used by Universal Command Server requires the user to provide a OS/400 user name / ID and a password. The Universal Command Server passes the name / ID and password to the operating system for verification; this is referred to as logging on the user.

If the user name and password are successfully validated by the operating system, the Initiator program (**UCMSINIT**) changes the current user profile to the user profile of the user ID.

## 11.6.4 Universal Access Control List

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

See Section 2.8 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 Command configuration file. See Section 2.2.6 Configuration File Syntax for detailed syntax information.

Table 11.3 identifies all Universal Command for OS/400 UACL entries. Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Command 3.2.0 Reference Guide.

| UACL Entry Name | Description |
|---|---|
| UCMD_ACCESS | Allows or denies access to Universal Command Server services |
| UCMD_REQUEST | Allows or denies access to Universal Command Server services based on client identification and request type |

Table 11.3 Universal Command Server for OS/400 - UACL Entries

# UACL Entry Precedence

## Deny or Allow Access

The **ucmd_access** rules are searched first for an entry that matches the client request. If an **ucmd_access** entry is found and the rule denies access to the Manager, the search stops and the Manager request is denied.

- If no **ucmd_access** entry is found or an **ucmd_access** rule allows access, the **ucmd_request** entries are searched. If an **ucmd_request** entry is found, its rule determines whether the Manager request is denied or allowed.
- If no rules are found, the Manager request is allowed.

## Authenticate or No Authenticate Access

The **ucmd_access** entries are searched followed by the **ucmd_request** entries.

- If an **ucmd_request** entry is found, it sets the authentication requirement.
- If no **ucmd_request** entry is found and an **ucmd_access** entry is found, the **ucmd_access** rule sets the authentication requirement.
- If no rules are found, the Manager request requires authentication.

Table 11.4, below, identifies the UACL entry precedence rules described above.

| ucmd_access | | ucmd_request | | Result |
|---|---|---|---|---|
| Allow/Deny | Auth/Noauth | Allow/Deny | Auth/Noauth | |
| NO-MATCH | NO-MATCH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| DENY | N/A | N/A | N/A | DENY |
| ALLOW | AUTH | NO-MATCH | NO-MATCH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | AUTH | ALLOW, AUTH |
| ALLOW | AUTH | ALLOW | NOAUTH | ALLOW, NOAUTH |
| ALLOW | AUTH | DENY | N/A | DENY |
| ALLOW | NOAUTH | NO-MATCH | NO-MATCH | ALLOW, NOAUTH |
| ALLOW | NOAUTH | ALLOW | AUTH | ALLOW,AUTH |
| ALLOW | NOAUTH | ALLOW | NOAUTH | ALLOW,NOAUTH |

Table 11.4 Universal Command for OS/400 - UACL Decision Table

## 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.

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

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

When no certificate is presented that maps to a certificate ID, the following set of rules effectively permit connections from any host, but has limited access from host 10.20.30.40 to user **TS1004** on that host. No host can execute commands as local user **root**. User **TS1004** on host 10.20.30.40 can execute commands as local user **tsup1004** without providing the password. Users **TS1004** from host 10.20.30.40 can execute commands as any local user by providing the local user password.

When a certificate is presented that maps to a certificate ID, certificate ID joe can request local user ID **tsup1004** without a password. Certificate ID **joe** is allowed to execute commands with any other local user ID with a password. Certificate ID operations cannot run anything. All other certificate IDs can execute commands with any user ID except for root with a password.

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

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

# Chapter 12
# Universal Command Server for HP NonStop

## 12.1 Overview

This chapter documents the Universal Command (UCMD) Server at a detailed level. The material is specific to the HP NonStop variety of operating systems.

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

### 12.1.1 Server Environment

The UCMD Server runs as a background process within the Open System Services (OSS) environment. It does not interact with a console.

All components dependent upon Universal Broker (such as UCMD Server) inherit the message language from the Universal Broker. All messages issued by components are sent to the Universal Broker for writing.

# 12.2 User Command Environment

The user request is executed by the UCMSINIT TACL script, which is started by the UCMD Server.

## 12.2.1 Universal Command Server Initiator

The UCMD Server Initiator is the TACL command executed by the UCMD Server. The Initiator executes the specific user command or script, captures the TACL completion code for the executed user command, and then returns the completion code of the user's command.

## 12.2.2 Command Shell

Since the UCMD Server process executes as an OSS process, the command shell used to execute the user's job is **`/bin/gtacl`**. The **`gtacl`** OSS program is used to execute the UCMD Server Initiator TACL script, UCMSINIT.

## 12.2.3 User Command Exit Code

The exit code returned to the UCMD Manager indicates the success or failure of the user-requested command. The UCMD Server returns the exit code to the UCMD Manager. The UCMD Server returns the exit code that it receives from the Initiator. The Initiator sets its exit code based on the completion code of the user commands or scripts. The Initiator then abends with this completion code, which is then propagated, back to the UCMD Manager process.

## 12.2.4 User Identification

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

- With user security active, the UCMD Server requires the UCMD Manager to supply a valid user ID for the local system and a password. The user command executes with the user ID and the primary and secondary group ID's of the user.
- With user security inactive, the UCMD Server does not require the UCMD Manager to supply a valid user ID. The user command executes with the user account of the UCMD Server. The user account of the UCMD Server is the super.super account.

## 12.2.5 Working Directory

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

- With user security active, a user command's working directory is the home directory of the user ID as defined in the **`/etc/passwd`** file.
- With user security inactive, a user command's working directory is the UCMD Server's working directory. All user commands executed use the same directory. Care should be taken to avoid name clashes and other consequences of multiple processes sharing a working directory.

## 12.2.6 Environment Variables

Environment variables are inherited from the server, which in turn inherits them from the broker.

If security is active, the following variables are modified to match the user environment: HOME, LOGNAME, USER, PWD, and SHELL. Their values are updated to reflect the values for the new environment.

The following variables are added if not found in the environment: HOME, USER, SHELL, and UCMDENV. The UCMDENV variable is set to a value of *1*. It can be used within scripts to determine if Universal Command has invoked them.

The UCMD Manager LOGIN option and UCMD Server LOGIN option have an impact on the environment variables defined. For login environments, the user's shell is invoked as a login shell. This, in turn, uses the shell profile file in the user's home directory. So any environment variables set in the profile file also will be defined. The UCMD Server inherits its environment variables from the Universal Broker. The user command, in turn, inherits its environment variables from the UCMD Server.

If user security is inactive, no changes are made to the environment variables.

# 12.3 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 UCMD Server for HP NonStop component definition is located in the component definition directory of the Universal Broker.

Table 12.1, below, identifies all of the options that comprise the UCMD Server for HP NonStop component definition.

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

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

Table 12.1 UCMD Server for HP NonStop - Component Definition Options

# 12.4 Configuration

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

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

## 12.4.1 Configuration File

The configuration file provides the simplest method of specifying configuration values that will not change with each command invocation.

The UCMD Server configuration file name is specified in the UCMD Server component definition. The default name is UCMDSCFG. See the component definition file to determine the subvolume in which it is located. This file can be edited manually using the TACL EDIT command.

## 12.4.2 Manager Override

A UCMD Manager can specify certain UCMD Server configuration options when it makes its request for command execution to the UCMD Server. The UCMD Manager SERVER_OPTIONS (-server) option is used to specify UCMD Server options.

Which options are available for UCMD Manager override depend on the UCMD Server platform and release. Chapter 3 Universal Command Server Configuration Options of the Universal Command 3.2.0 Reference Guide identifies which UCMD Server for HP NonStop configuration options (see Table 12.2) for which manager override is applicable.

The UCM Manager is not notified of override errors. The UCMD Server logs the error and continues processing the request.

## 12.4.3 Configuration Options Summary

Table 12.2, below, identifies all UCMD Server for z/OS configuration options. Each **Option Name** is a link to detailed information about that configuration option in the Universal Command 3.2.0 Reference Guide.

| Option Name | Description |
|---|---|
| CODE_PAGE | Code page used for text translation. |
| CPU | Number of processor on which job is to run. |
| DATA_AUTHENTICATION | Specification for whether or not data integrity checks are performed on all standard I/O files. |
| DATA_COMPRESSION | Specification for whether or not data is compressed on all standard I/O files. |
| DATA_ENCRYPTION | Specification for whether or not data is encrypted on all standard I/O files. |
| INSTALLATION_DIRECTORY | Base directory in which the product is installed. |
| JOB_RETENTION | Number of seconds a disconnected server remains active after user process completes. |
| KEEPALIVE_INTERVAL | Specifies if and how often a keepalive message is sent. |
| LOGIN | Setup and login environment or not. |
| MESSAGE_LEVEL | Level of messages written. |
| NETWORK_FAULT_TOLERANT | Specifies whether or not the server accepts the network fault tolerant protocol. |
| PRIORITY | Execution priority of the user job being run. |
| SCRIPT_TYPE | Script type of the user job being run. |
| USE_USER_ACCOUNTING_CODE | Specifies whether or not user authentication is active. |

Table 12.2 UCMD Server for HP NonStop - Configuration Options

# 12.5 Security

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

Universal Command security concerns are:

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

## 12.5.1 File Permissions

Only trusted user accounts should have write permission to the UCMD Server installation subvolume and all files within them.

## 12.5.2 Configuration Files

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

## 12.5.3 Universal Command Server User ID

UCMD Server requires read access to its installation subvolume. If user security is activated, the UCMD Server requires super.super access to create processes that execute with another user's identity. The UCMD Server security identity is inherited from the Universal Broker. If the Universal Broker is running with a non-super.super user ID, the UCMD Server program must have the ProgID bit set and super.super as owner. (See the Universal Message Translator chapter in the Universal Products Utilities 3.2.0 User Guide for details.)

## 12.5.4 User Authentication

User authentication is the process of verifying that a user is a known and valid user. The process used by UCMD Server requires the user to provide an HP NonStop user name / ID and a password. The UCMD 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 Command uses Default user authentication for traditional password comparison method.

## 12.5.5 Universal Access Control List

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

See Section 2.8 Universal Access Control List for details on the UACL feature.

### UACL Entries

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

Table 12.3 identifies all Universal Command Server for HP NonStop UACL entries. Each **UACL Entry Name** is a link to detailed information about that UACL entry in the Universal Command 3.2.0 Reference Guide.

| UACL Entry Name | Description |
|---|---|
| UCMD_ACCESS | Allows or denies access to Universal Command Server services |

Table 12.3 Universal Command 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.

```
ucmd_access     10.20.30.,*,*,allow,auth
ucmd_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.

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

# Chapter 13
# User Scripts

## 13.1 Overview

This chapter provides examples of user scripts to be used with Universal Command.

They are provided here to help you get the most of your Universal Command deployment. If you have code or scripts that you think will help the Universal Command user community, please email them to Stonebranch, Inc. Customer Support.

Although Stonebranch reviews submissions to assure their basic functionality, it cannot guarantee that they will work satisfactorily in your environment. Likewise, Stonebranch cannot provide customer support for these examples, since they can be modified outside of Stonebranch control.

# 13.2 Verify Disk Space

The following scripts can be executed throughout your processing to verify that the amount of disk space required is available.

## 13.2.1 UNIX Example

Figure 13.1, below, provides an example of a Verify Disk Space user script for UNIX.

```
#!/bin/sh
#
# DATE          : August 2001
# AUTHOR        : Colin Cocksedge
# DESCRIPTION   : Example script to find out if the space required is
#               : Available.
#               : Usage ./space? -m mountpoint -s required space -o space
#               : operator.
#.              : NB: space operator should be k (for kilobytes),
#               : m (for megabytes) or g (for gigabytes).
#               : Script returns 0 if space is available or 8 if not.
#               : SOLARIS requires -x switch which changes the parm
#               :         for the df command.
# ENVIRONMENTS  : Currently tested on the following UNIX variants -
#               : USS 2.10 - AIX 4.2 & 4.3 - HPUX 1020 & 1100
#               : LINUX - SOLARIS - RELIANT

freesp=0
mountp=.
oper=.
sapce=.
parm=-kP

# read options
while getopts m:s:o:xvh opt
```

```
do
    case $opt in
      m) mountp=$OPTARG;;
      s) space=$OPTARG;;
      o) oper=$OPTARG;;
      x) parm="-k";;
      v) echo "version = 1.0 08/2001"
         exit 0;;
      h)
    echo \
    "usage: $0 .-m mountpoint. .-s space required. .-o space operator k/m/g."
    exit 0;;
      \?)
    echo >&2 \
    "usage: $0 .-m mountpoint. .-s space required. .-o space operator k/m/g."
    exit 12;;
    esac
done
     shift `expr $OPTIND - 1`

# check parms supplied
if [ $mountp = . ]; then
   echo \
   "-m mountpoint - required parameter missing"
   exit 12
fi
if [ $space = . ]; then
   echo \
   "-s space required - required parameter missing"
   exit 12
fi
if [ $oper = . ]; then
   echo \
   "-o space operator k/m/g - required parameter missing"
   exit 12
fi

# find out how much space
freesp=`df $parm $mountp | awk 'NR == 2 {print $4}'`
rc=$?
if [ $rc -ne 0 ]; then
   exit $rc
fi
```

```
# multiply available space by relevant operator
if [ $oper = m ]; then
   freesp=`expr $freesp / 1024`
fi
if [ $oper = g ]; then
   freesp=`expr $freesp / 1048576`
fi

echo  "Required space = "$space""$oper" - Actual Space = "$freesp""$oper

# compare actual to required
if [ $freesp -ge $space ]; then
   echo  "Required space exists for "$mountp" .. exiting rc=0"
   exit 0
else
   echo  "Not enough space exists for "$mountp" .. exiting rc=8"
   exit 8
fi

exit 12
```

Figure 13.1 Verify Disk Space - UNIX

## JCL

Figure 13.2, below, illustrates how to execute the script in Figure 13.1 from a z/OS Universal Command Manager. The script also can be executed outside of Universal Command. In this example, the script is housed on the z/OS.

```
//UCMDSTEP EXEC UCMDPRC
//SCRIPTDD DD DISP=SHR,DSN=hlq.UNV.SUNVUSRS(USCRP001)
//SYSIN DD *
-script SCRIPTDD -options "-m /usr/local/job -s 1 -o g" -host dallas
-userid joe -pwd abcdefg
/*
```

Figure 13.2 JCL for Verify Disk Space - 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 of the command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to SYSOUT in the **UCMDPRC** procedure.

The process will be authenticated to the remote machine with User ID **joe** and password **abcdefg**. The process will run under the authority of User ID **joe**. The script will return a *0* if the space is available; it will return an *8* if the space is not available.

## SYSIN Options

The SYSIN options used are:

| SYSIN Options | Description |
|---|---|
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |
| -options | Specifies the command line options to pass to the script file |
| -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. |

## Script Space? Options

The script space? options used are:

| Options | Description |
|---|---|
| -m | Specifies the mount point to check |
| -s | Specifies the size to check for. |
| -o | Specifies the space measurement. |

## 13.2.2 Windows Example

Figure 13.3, below, provides an example of a Verify Disk Space user script for Windows.

```
/* REXX */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* MEMBER      : USCRP002 REXX PROCEDURE                              */
/*                                                                    */
/* DESCRIPTION : SAMPLE REXX PROCEDURE TO INVOKE UNIVERSAL COMMAND    */
/* .           : WITH SPECIFIED DIR CMD AND VERIFY THAT ADEQUATE      */
/* .           : SPACE IS AVAILABLE ON THE SPECIFIED NT DRIVE.        */
/*                                                                    */
/* SETUP       : USE WITH JCL MEMBER USCRPJ02                         */
/*                                                                    */
ARG HLQ RSPACE OPER
"CALL '"HLQ".UNV.SUNVLOAD(UCMD)'"
"EXECIO * DISKR UNVOUT (FINIS"
   DO WHILE QUEUED() > 0
      PULL A
       CHKWORD = SUBWORD(A,5,1)
       IF CHKWORD = "FREE" THEN DO
          FREESP = SUBWORD(A,3,1)
          SAY "*** FREESPACE IS "FREESP
          SAY "*** REQUIRED SPACE IS "RSPACE
          IF FREESP > RSPACE THEN
             RC = 0
             ELSE RC = 8
       END
    END
 EXIT RC
```

Figure 13.3 Verify Disk Space - Windows

## JCL

Figure 13.4, below, illustrates how to execute the above script from a z/OS Universal Command Manager.

```
//stepname  EXEC PGM=IKJEFT01
//STEPLIB   DD DISP=SHR,DSN=hlq.UNV.SUNVLOAD
//UNVNLS    DD DISP=SHR,DSN=hlq.UNV.SUNVNLS
//SYSLBC    DD DISP=SHR,DSN=SYS1.BRODCAST
//SYSPROC   DD DISP=SHR,DSN=hlq.UNV.SUNVUSRS
//* SPECIFY DIR COMMAND WITH DRIVE LETTER REQUIRED.
//SYSIN     DD *
 -host dallas -userid joe -pwd password -c "DIR c:\"

//UNVIN     DD DUMMY
//UNVOUT    DD DISP=(NEW,DELETE),DSN=&&UNVOUT,
//             SPACE=(CYL,(1,1)),UNIT=SYSDA,VOL=SER=volser,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//UNVERR    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTERM   DD SYSOUT=*
//* SPECIFY HLQ FOR UCMD DATASETS AND SPACE REQUIRED IN BYTES.
//SYSTSIN   DD *
  %USCRP002 HLQ 5,000,000,000
/*
```

Figure 13.4 JCL for Verify Disk Space - Windows

The JCL is used to execute the REXX program USCRP002. The JCL must be modified to change HLQ to the HLQ of your Universal Command installation. The VOL=SER= value must be changed to a volume at your site.

The command is sent to a remote system named **dallas** for execution. The output of the command is redirected back to the Universal Command batch job and written to ddname UNVOUT.

The process will be authenticated to the remote machine with userid of **joe** and password of **normal**. The process will run under the authority of userid **joe**. The script will return a *0* if the space is available; it will return an *8* if the space is not available.

## SYSIN Options

The SYSIN options used are:

| SYSIN Options | Description |
|---|---|
| `-c` | Specifies the Windows **`Dir`** command, as well as the directory to check for available space. |
| `-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. |

## REXX Program via DD SYSTIN Options

The options for REXX program via DD SYSTSIN are:

| Options | Description |
|---|---|
| `%` | REXX program name. This program must reside in the library defined in the SYSPROC DD statement. |
| `HLQ` | Change this parameter to the HLQ of the Universal Command Install files. |
| `5,000,000,000` | Specifies the amount of space required in bytes. |

# 13.3 Verify Task Status

The following script can be executed to determine if a service is active or not.

## 13.3.1 Windows Example

Figure 13.5, below, provides an example of a Verify Task Status user script for Windows.

```
/* REXX */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* MEMBER      : USCRP003 REXX PROCEDURE                              */
/*                                                                    */
/* DESCRIPTION : SAMPLE REXX PROCEDURE TO INVOKE UNIVERSAL COMMAND    */
/* .           : WITH NET START COMMAND AND VERIFY A SPECIFIC SERVICE */
/* .           : IS ACTIVE.                                           */
/*                                                                    */
/* SETUP       : USE WITH JCL MEMBER USCRPJ03                         */
/*                                                                    */
ARG HLQ SERVICE
ACTIVE = NO
"CALL '"HLQ".UNV.SUNVLOAD(UCMD)'"
"EXECIO * DISKR UNVOUT (FINIS"
   DO WHILE QUEUED() > 0
       PULL A
       IF SERVICE = A THEN DO
       ACTIVE = YES
       END
    END
 IF ACTIVE = YES THEN DO
    SAY SERVICE is Running
    RC = 0
    END
    ELSE DO
    SAY SERVICE is Not Running
    RC = 8
    END
 EXIT RC
```

Figure 13.5 Verify Task Status - Windows

## 13.3.2 JCL

Figure 13.6, below, illustrates how to execute the Figure 13.5 user script from a z/OS Universal Command manager.

```
//stepname    EXEC PGM=IKJEFT01
//STEPLIB   DD DISP=SHR,DSN=hlq.UNV.SUNVLOAD
//UNVNLS    DD DISP=SHR,DSN=hqq.UNV.SUNVNLS
//SYSLBC    DD DISP=SHR,DSN=SYS1.BRODCAST
//SYSPROC   DD DISP=SHR,DSN=hqq.UNV.SUNVUSRS
//* Issues NET START COMMAND
//SYSIN     DD *
 -host dallas -userid joe -pwd password -c "NET START"

//UNVIN     DD DUMMY
//UNVOUT    DD DISP=(NEW,DELETE),DSN=&&UNVOUT,
//             SPACE=(CYL,(1,1)),UNIT=SYSDA,VOL=SER=volser,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//UNVERR    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTERM   DD SYSOUT=*
//* Specify UCMD HLQ &
//* Specify Service your are checking.
//* Issue NET START on NT to find out the service name to search for
//SYSTSIN   DD *
  %USCRP003 HLQ Norton Antivirus Service
```

Figure 13.6 JCL for Verify Task Status - Windows

The JCL is used to execute the REXX program NETSTART. The JCL must be modified to change HLQ to the HLQ of your Universal Command installation. The VOL=SER= value must be changed to a volume at your site.

The command is sent to a remote system named **dallas** for execution. The output of the command is redirected back to the Universal Command batch job and written to ddname UNVOUT.

The process will be authenticated to the remote machine with userid of **joe** and password of **normal**. The process will run under the authority of userid **joe**. The script will return a *0* if the service is active; it will return an *8* if the service is not active.

## SYSIN Options

The SYSIN options used are:

| SYSIN Options | Description |
|---|---|
| -c | Specifies the Windows **Dir** command, as well as the directory to check for available space. |
| -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. |

## REXX Program via DD SYSTIN Options

The options for REXX program via DD SYSTSIN are:

| Options | Description |
|---|---|
| % | REXX program name. This program must reside in the library defined in the SYSPROC DD statement. |
| HLQ | Change this parameter to the HLQ of the Universal Command Install files. |
| Norton antivirus service | Replace with the service name to check if it is active. |

# 13.4 Network Status Report

The following script can be executed to determine the status of the network.

## 13.4.1 UNIX Example

Figure 13.7, below, illustrates a script that produces a report of the system status of a remote UNIX system. Instead of executing a command on the remote host, a local script file is executed.

Note that the commands executed in the script file may or may not require modifications, depending on the type of UNIX system on which it executes.

```
echo "System Status as of `date`"
echo "-------------------------------------------------"
netstat
echo "-------------------------------------------------"
df
echo "-------------------------------------------------"
ps –ax
```

Figure 13.7 Network Status Report script - UNIX

### JCL

Figure 13.8, below, illustrates how to execute the Figure 13.7 script from a z/OS Universal Command manager.

```
//stepname EXEC UCMDPRC
//MYSCRIPT DD DISP=SHR,DSN=hlq.UNV.SCRIPTS(USCRP004)
//SYSIN DD *
-script myscript -host dallas -userid joe –pwd akkSdiq
```

Figure 13.8 JCL for Network Status Report Script - UNIX

The JCL procedure UCMDPRC is used to execute the command. The report is written to the remote command's standard out DD UNVOUT that is allocated to SYSOUT in the UCMDPRC procedure. The command options execute the script allocated on DD MYSCRIPT.

## SYSIN Options

The SYSIN options used are:

| SYSIN Options | 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`**. |
| `-userid` | Specifies the remote user ID with which to execute the command. |
| `-pwd` | Specifies the password for the user ID. |

# 13.5 File Watchers

The following scripts can be executed to determine if a file exists.

## 13.5.1 Perl Script: Watch for One File

Figure 13.9, below, illustrates a script that can be used to wait for one file at a time. It can be executed in batch either at the server or from a Universal Command manager job. The length of time that the script will stay active is determined by a parameter passed to the script at execution time.

This script has been written and tested with Perl 5.6.1.

Windows and UNIX Perl interpreters are available from:

www.perl.com/pub/a/language/info/software.html

```
#!/usr/bin/perl
#
# Purpose:
# Will check for existence of a given file.
# If file exists script waits for 1 minute and if file size is
# unchanged returns 0, if filesize has changed will continue comparing
# filesize every 1 minute until either file size is unchanged (returns 0)
# or until time is up (returns 4). If file does not exist will continue to
# check every 1 minute up to the limit of the time specified (returns 8).
#
# Usage:
# USCRP005 file=filename time=time
# Where filename = name of file to find (including path if required), time is
# time to search in minutes or time of day to stop searching in format hh:mm
# (24 hour format). If time of day used then next occurrence is used,
# i.e. will not wait more than 24 hours. If no time specified defaults to
# duration 10 minutes.
# If filename or path contains spaces use ! instead of space i.e.
# c:\program files\file.name should be c:\program!files\file.name
```

```
#
# Return Code Conventions used:
# 0  = File Found.
# 4  = File may not be complete.
# 8  = File not found.
# 12 = Parameter Error.
# 16 = Severe Error.
#
# Environment:
# Written and tested with Perl 5.6.1
# Windows and Unix Perl interpreters are available from www.perl.com/pub.
# Tested in the following environments:
#  Windows 2000 Professional
#  HPUX 1100
# Should work with any Windows or Unix platform.
#
# Version:
$vers = '1.0 05/25/2001';


# select statements to unbuffer stdout and stderr

select(STDERR); $| = 1;
select(STDOUT); $| = 1;

# get local time and sort out format

($x,$mins,$hour,$mday,$mnth,$year,$x,$x,$x) = localtime(time);
$year += 1900;
$mnth = ($mnth + 1);
if (length $mins eq 1) {
    $mins = join('','0',$mins);
}

# get login id

$user = getlogin;

# output initialization information

print "wait4file $vers starting on $mnth/$mday/$year at $hour:$mins \n";
print "login userid = $user \n";
```

```
# set default variables

$file = 'no.file';
$time = 'no.time';
$last = '0';

# read and verify input parms

my $arg;
my %args;

foreach $arg (0..$#ARGV) {
    my ($name,$value) = split(/=/,$ARGV[$arg]);
    $file = "$value" if ($name eq 'file');
    $time = "$value" if ($name eq 'time');
    $invl = "$value" if ($name eq 'invl');
}

if ($file eq 'no.file') {
    print "ERROR*** No file name specified \n";
    exit 12
}
else {
    $spaces = index $file,'!';
    if ($spaces ne -1) {
        $file =~ s|!| |g;
    }
    print "Searching for file = $file \n";
}

if ($time eq 'no.time') {
    $time = 10;
    print "No time specified - will use default 10 minutes \n";
}
else {
    $test = '0';
    $test = index $time,':';
    if ($test eq 2) {
   if (length $time ne 5) {
       print "ERROR*** $time - format invalid \n";
       exit 12
```

```
   }
        $save = $time;
   $shrs = substr $time,0,2;
   $smin = substr $time,3,2;
   $stim = join('',$shrs,$smin);
   print "Will search until $time \n";
    }
    else {
   print "Will search for $time minutes \n";
    }
}

# subroutine to check time

sub check_time {
    ($x,$mins,$hour,$x,$x,$x,$x,$x,$x) = localtime(time);
    if (length $mins eq 1) {
        $mins = join('','0',$mins);
    }
    $rtim = join('',$hour,$mins);
    if ($rtim eq $stim) {
        $time = 0;
        print "It is now $save, time to stop waiting \n";
    }
    else {
        $time = 2;
    }
    return $time;
}
# check time if we need too

if ($test eq 2) {
    check_time();
}

# Loop to look for file

while ($time > 0) {
    $size = 'no.size';
    $size = -s $file;
    if ($size eq '') {
        print "$file not found yet .. still looking \n";
        $comp = "no";
    }
```

```
    else {
        if ($size eq $last) {
            print "$file exists .. mission accomplished \n";
            exit 0
        }
        else {
            print "$file exists .. still checking growth \n";
            $comp = "partial";
        }
        $last = $size;
    }
    sleep 60;
    $time = ($time - 1);
    if ($test eq 2) {
        check_time();
     }
}

# set return codes if we didn't complete already

if ($comp = "no") {print "$file not found within time limit \n"; exit 8}
if ($comp = "partial") {print "$file may not be complete !!! \n"; exit 4}

# oops something went wrong

exit 16
```

Figure 13.9 Perl Script - Watch for One File

## JCL

Figure 13.10, below, illustrates how to execute the Figure 13.9 script from a z/OS Universal Command manager. The script resides on the z/OS.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD DISP=SHR,DSN=hlq.UNV.SUNVUSRS(USCRP005)
//SYSIN    DD *
-script SCRIPTDD -host dallas -userid joe -pwd abcdefg
-options "file=filename time=06:00"
-script_type pl
/*
```

Figure 13.10 JCL for Perl Script - Watch for One File

The JCL is used to execute the Perl script USCRP005. The output of the command is redirected back to the Universal Command batch job and written to ddname UNVOUT. The script will check for existence of a given file. If file exists, the script waits for 1 minute and if file size is unchanged returns 0, if file size has changed it will continue comparing file size every 1 minute until either file size is unchanged (returns 0) or until time is up (returns 4). If file does not exist, it will continue to check every 1 minute up to the limit of the time specified (returns 8).

## SYSIN Options

The SYSIN options used are:

| Option | Description |
|---|---|
| -script | DD from which to read a script file. The script file is sent to the remote system for execution. |
| -options | Command line options to pass to the script file. |
| -host | Directs the command to a computer with a host name of dallas. |
| -userid | Remote user ID with which to execute the command. |
| -pwd | Password for the user ID. |
| -script_type | Type of script specified by the SCRIPT_FILE option. |

## Script USCRP005 Options

The options for script USCRP005 are:

| Option | Description |
|---|---|
| file= | Name of the file that is being checked for existence. |
| time= | Time to stop checking for existence in HH:MM format. If not coded, the script will check for 10 minutes. |

## 13.5.2 Perl Script: Watch for One or More Files and Perform Action

Figure 13.11, below, illustrates a script that can be executed to look for one or more files determined by an input file containing a list of file names. Once a file exists, a command is executed on the server. That command could be a Universal Command Manager to talk back to the OS/390 via USS. This script can be run as a UNIX daemon or a Windows service, or as batch job **/stc** from z/OS. It can be executed on the server or from within a Universal Command manager job on z/OS.

This script has been written and tested with Perl 5.6.1.

Windows and Unix Perl interpreters are available from:

www.perl.com/pub/a/language/info/software.html

```
#!/usr/bin/perl
#
# Purpose:
# Will execute a specified command when a specified file is found.
# Reads watch4file.conf to get a list of files to search for. When a
# file is found the script pauses for 20 seconds and double checks
# the size to confirm that the file is not growing. If the file is
# still growing it is skipped until the next cycle. If the file has
# not grown in this time then the command associated with the file
# is executed.
#
# It is recommended that the command specified is a script which performs
# a rename of the file as its first action to ensure that the file is
# not processed twice by this script.
#
# Messages are output to watch4file.log or stdout (depending upon
# startup parms).
#
# Usage:
# watch4file config=configpath log=stdout interval=seconds daemon=yes
# config= specifies the path of the conf and log files. If
# the path contains spaces use ! instead of space i.e.
# c:\program files\file.name should be c:\program!files\file.name
# log=stdout can be specified if messages should be written to stdout.
# if the log parameter is omitted the log file is used.
# interval= specifies an interval between cycles other than
# the default 60 seconds.
```

```
# daemon=yes specify this parameter if you are running this script as
# a unix daemon. Do not use log=stdout if using this option.
#
# Environment:
# Written and tested with Perl 5.6.1
# Windows and Unix Perl interpreters are available from www.perl.com/pub.
# Tested in the following environments:
#  Windows 2000 Professional
#  HPUX 1100
# Should work with any Windows or Unix platform.
#

# Version:
$vers = '0.1 06/04/2001';

# setup default vars
$interval = '60';

# read input parms
my $arg;
my %args;

foreach $arg (0..$#ARGV) {
    my ($name,$value) = split(/=/,$ARGV[$arg]);
    $config = "$value" if ($name eq 'config');
    $log = "$value" if ($name eq 'log');
    $interval = "$value" if ($name eq 'interval');
    $daemon = "$value" if ($name eq 'daemon');
}
# if we need to run as a daemon fork a new process
if ($daemon eq 'yes') {
    defined(my $pid = fork) or die "can't fork: $!";
    exit if $pid;
    setsid or die "can't start a new session: $!";
    umask 0;
}

$spaces = index $config,'!';
if ($spaces ne -1) {
    $config =~ s|!| |g;
}
```

```
if ($config ne '') {
    chdir $config or die "Can't cd to $config: $!\n";
}

# open logfile and select as default output for print
if ($log ne 'stdout') {
    open(LOGFILE, '>' , "watch4file.logfile");
    select(LOGFILE); $| = 1;
    open(STDOUT, ">&LOGFILE");
    open(STDERR, ">&LOGFILE");
}
else {
    select(STDOUT); $| =1;
    open(STDERR, ">&STDOUT");
}

# get current date and time and sort out formats
($x,$mins,$hour,$mday,$mnth,$year,$x,$x,$x) = localtime(time);

$year += 1900;
$mnth = ($mnth + 1);
if (length $mins eq 1) {
    $mins = join('','0',$mins);
}

# get the current user
$user = getlogin;

# print initialization info to logfile
print "watch4file $vers starting on $mnth/$mday/$year at $hour:$mins \n";
print "Login userid = $user \n";
print "Process id   = $$ \n \n";
print "Will use config path $config \n";
print "Interval between cycles is $interval seconds \n \n";

# this is the main loop - loop forever.
while () {
    # timestamp the start of this cycle
    ($x,$mins,$hour,$mday,$mnth,$year,$x,$x,$x) = localtime(time);
    $year += 1900;
    $mnth = ($mnth + 1);
    if (length $mins eq 1) {
        $mins = join('','0',$mins);
    }
```

```
    print "new cycle starting at $hour:$mins on $mnth/$mday/$year \n";

    # read the config file
   open(CONFIG, '<' , "watch4file.conf") or print "can't open config file : $!
\n";
    local $_;
    while (<CONFIG>) {
        # ignore comments

   next if /^#/;
   ($file,$cmd) = split(/<==>/,$_);
        # look for the file
   print "==> searching for file $file \n";
   $size = 'no.size';
   $size = -s $file;
        if ($size eq '') {
            print "    not found yet \n";
            next;
        }
        else {
            print "    file exists .. checking growth \n";
       $last = $size;
       sleep 20;
       $size = -s $file;
       if ($size eq $last) {
           print "    exists .. executing $cmd \n";
           system($cmd) == 0 or print "*** $! $? \n";
           print "    ...cmd complete $? \n";
       }
            else {
           print "    still growing will recheck next cycle \n";
       }
   }
    }
    close CONFIG;
    print "cycle complete \n \n";
    sleep $interval;
}
```

Figure 13.11 Perl Script - Watch for One or More Files and Perform Action

## JCL

Figure 13.12, below, illustrates how to execute the Figure 13.11 script from a z/OS Universal Command manager.

```
//stepname EXEC UCMDPRC
//SCRIPTDD DD DISP=SHR,DSN=hlq.UNV.SUNVUSRS(USCRP006)
//SYSIN    DD *
-script SCRIPTDD -host dallas -userid joe -pwd abcdefg
-options "config=c:\temp log=stdout"
-script_type pl
/*
```

Figure 13.12 JCL for Perl Script - Watch for One or More Files and Perform Action

The JCL is used to execute the Perl script USCRP006. The output of the command is redirected back to the Universal Command batch job and written to ddname UNVOUT. The script will execute a specified command when a specified file is found. It reads **watch4file.conf** in the **config** directory to get a list of files for which to search. When a file is found, the script pauses for 20 seconds and double checks the size to confirm that the file is not growing. If the file is still growing, it is skipped until the next cycle. If the file has not grown in this time, the command associated with the file is executed.

## SYSIN Options

The SYSIN options used are:

| Option | Description |
|---|---|
| -script | DD from which to read a script file. The script file is sent to the remote system for execution. |
| -options | Command line options to pass to the script file. |
| -host | Directs the command to a computer with a host name of dallas. |
| -userid | Remote user ID with which to execute the command. |
| -pwd | Password for the user ID. |
| -script_type | Type of script specified by the SCRIPT_FILE option. |

## PERL Script Options

The options for PERL script are:

| Options | Description |
|---|---|
| config= | Name of the directory that contains the config file (**watch4file.conf**). |
| log= | Location of the output. If omitted, records are written to **watch4file.log** in the **config** directory. |

## Configuration File

Figure 13.13, below, illustrates a sample configuration file.

```
#
# Configuration file format:
# Filename<==>Command
# where:
# Filename = the name of the file to search for
# Command  = the command to execute when the file is found
#
# It is recommended that you rename the file as soon as it is found to # avoid
processing the same file twice.
#
c:\testfile.txt<==>c:\testfilescript.bat
/testfile2.txt<==>./testfile2script
c:\testfile3.txt<==>c:\testfile3script.bat
```

Figure 13.13 Configuration File

Figure 13.14, below, illustrate the contents of a script to execute when the file is found.

```
REM The following lines are a Windows .bat file example:
REM The ucmd command should be on 1 line or use the NT continuation ¬

rename c:\testfile.txt c:\testfileold.txt
ucmd -c " /opt/universal/bin/uwto -msg 'testfile.txt is here'"
 -host dallas -userid joe -pwd abcdefg


#!/bin/sh

# The following is a shell script example for UNIX
# The ucmd command should be on 1 line or use the unix continuation \
#
#
mv /testfile.txt /testfileold.txt
cd /opt/universal/bin
ucmd -c "/opt/universal/bin/uwto -msg 'testfile.txt is here'"
 -host dallas -userid joe -pwd abcdefg
```

Figure 13.14 Script to Execute

## Execute as a UNIX Daemon

Figure 13.15, below, illustrates how to execute the Figure 13.14 script as a UNIX daemon.

```
c:\watch4file.pl config=c:\temp daemon=yes
```

Figure 13.15 Execute Script as UNIX Daemon

The output of the command is redirected back to the **watch4file.log** located in **c:\temp** directory. The script will execute a specified command when a specified file is found. It reads **watch4file.conf** in the **config** directory to get a list of files for which to to search. When a file is found the script pauses for 20 seconds and double checks the size to confirm that the file is not growing. If the file still is growing, it is skipped until the next cycle. If the file has not grown in this time, the command associated with the file is executed.

### PERL Script Options

The options for PERL script are:

| Option | Description |
|---|---|
| config= | Name of the directory that contains the config file. |
| daemon= | Specification for whether or not the process is a UNIX daemon. |

## 13.5.3 REXX Script: Watch for One or More Files and Perform Action

Figure 13.16, below, illustrates a script that is executed from the z/OS system to look for one or more remote files determined by an input file containing a list of file names and remote servers.

Once a file exists, a command is executed on the z/OS system. This script can be run as a batch job or stc from z/OS.

```
/* REXX */
/* ********************************************************************/
/* (c) Copyright 2000-2008, Stonebranch, Inc.   All rights reserved.  */
/*                                                                    */
/* Stonebranch, Inc.                                                  */
/* REXX Script to check for existence of file and perform action.     */
/* Version 2 Release 2 Modification 0                                 */
/*                                                                    */
/* SYSMOD:    TUCM220                                                 */
/* Type:      USRS                                                    */
/* Name:      USCRP007                                                */
/*                                                                    */
/* Disclaimer for all User Written Script Examples                    */
/* ----------------------------------------------                     */
/*                                                                    */
/* Although Stonebranch reviews submissions to assure their basic     */
/* functionality, Stonebranch cannot gaurantee that they              */
/* will work satisfactorily in your environment, nor can              */
/* Stonebranch provide technical support for these examples           */
/* since they can be modified outside of Stonebranch control.         */
/*                                                                    */
/* Description                                                        */
/* -----------                                                        */
/* This script is executed from the z/OS system to look for 1 or more */
/* remote files determined by an input file containing a list of file */
/* names on remote servers.  Once a file exists, a command is         */
/* executed on the z/OS system. This script can be run as a batch job */
/* or a stc.                                                          */
/*                                                                    */
/* Member USCRPJ07 is an example of the JCL used to execute this      */
/*       script.  See this member for implementation instructions.    */
/* Member FWCONFDD is an example of the configuration file.           */
/* Member FWPARMDD is an example of a the parm file containing the    */
/*       files and servers to look for.                               */
/*                                                                    */
```

```
/*-------------------------------------------------------------------*/
TRACE OFF
/*-------------------------------------------------------------------*/
/*                                                                   */
/* MEMBER      : FW REXX PROCEDURE                                   */
/*                                                                   */
/* DESCRIPTION : SEE MEMBER #README                                  */
/*                                                                   */
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/* MAIN PROCEDURE                                                    */
/*-------------------------------------------------------------------*/
CALL FW.CONF
IF REXX.TRACE = 'YES' THEN TRACE R
CALL FW.PARM
FW.LOOP:
  CALL FW.CYCL
  CALL FW.WAIT
  SIGNAL FW.LOOP
FW.EXIT:
  EXIT FW.CC
/* FW.CONF ---------------------------------------------------------*/
/* INITIAL TASKS EXECUTES ONLY ONCE                                  */
/*  1 - ALLOCATE FILES                                               */
/*  2 - WRITE INITIALIZATION MESSAGES                                */
/*  3 - ASSIGN CONFIGURATION PARMS TO VARIABLES                      */
/*  4 - SETUP SYSCALL ENVIRONMENT FOR SLEEP FUNCTION                 */
/*-------------------------------------------------------------------*/
FW.CONF:
  FW.VN = '1.0.0 - 040131'
  FW.CC = 0
  HC = 0
  SUNVLOAD = LISTDSI("UNVLOAD" "FILE")
  PARSE VALUE DATE('U') WITH DATE
  PARSE VALUE TIME('C') WITH TIME

  PUSH "FW-I01 VERSION "FW.VN" - EXECUTING ON "DATE" AT "TIME
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
```

```
  PARSE SOURCE . . . . . ENVIR .
  IF ENVIR \= 'SH' THEN ,
    IF SYSCALLS('ON')>3 THEN
      DO
      PUSH "FW-XXX ERROR UNABLE TO ESTABLISH THE SYSCALL ENVIRONMENT"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      FW.CC = '12'
      SIGNAL FW.EXIT
      RETURN
    END

  PUSH "FW-I02 READING CONFIGURATION PARAMETERS"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  "EXECIO * DISKR FWCONFDD (FINIS"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR READING CONFIGURATION FROM FWCONFDD"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END

ELSE DO WHILE QUEUED() > 0
    PARSE PULL PARMNAME '=' PARMVALU
    IF SUBWORD(PARMNAME,1,1) = "#" THEN ITERATE
    IF SUBWORD(PARMNAME,1,1) = " " THEN ITERATE
    PARMVALU=STRIP(PARMVALU)
    PARMNAME=STRIP(PARMNAME)
    PARMNAME=VALUE(PARMNAME,PARMVALU)
    PUSH "  -   " PARMNAME "=" PARMVALU
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
  END
IF REXX.TRACE = 'REXX.TRACE' THEN REXX.TRACE = 'NO'
IF UCMD.TRACE = 'UCMD.TRACE' THEN UCMD.TRACE = 'ERROR'
IF MSG.PREFIX = 'MSG.PREFIX' THEN MSG.PREFIX = ' '
IF CYCLE.INTERVAL = 'CYCLE.INTERVAL' THEN CYCLE.INTERVAL = '60'

RETURN

/* FW.PARM ------------------------------------------------------------*/
/* READ PARAMETERS AND PUT THEM IN THE STACK                           */
/*--------------------------------------------------------------------*/
```

```
FW.PARM:
  PUSH "FW-I03 READING PARAMETER FILE"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  "EXECIO * DISKR FWPARMDD (FINIS"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR READING PARAMETERS FROM FWPARMDD"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
DO WHILE QUEUED() > 0
    PARSE PULL FW.PARM
    IF SUBSTR(FW.PARM,1,1) = "#" THEN ITERATE
    IF SUBSTR(FW.PARM,1,1) = " " THEN ITERATE
    IF SUBSTR(FW.PARM,1,4) = "HOST" THEN CALL FW.HOST
    IF SUBSTR(FW.PARM,1,4) = "FILE" THEN CALL FW.FILE
    IF SUBSTR(FW.PARM,1,6) = "RENAME" THEN CALL FW.RENM
    IF SUBSTR(FW.PARM,1,11) = "CHECKGROWTH" THEN CALL FW.CHKG
    IF SUBSTR(FW.PARM,1,6) = "ACTION" THEN CALL FW.ACTN
    IF SUBSTR(FW.PARM,1,3) = "END" THEN CALL FW.HEND
  END
RETURN

/* FW.CYCL ------------------------------------------------------------*/
/* TRY TO FIND THE FILES                                               */
/*--------------------------------------------------------------------*/
FW.CYCL:
  PARSE VALUE DATE('U') WITH DATE
  PARSE VALUE TIME('C') WITH TIME
  PUSH "FW-I05 NEW CYCLE STARTING ON" DATE "AT" TIME
  "EXECIO 1 DISKW FWLOGFDD (FINIS"

  DO HCC = 1 TO HC
    CALL FW.ALOC
    CALL FW.WRTE.SYSIN
    IF TYPE.HCC = "WIN" THEN CALL WIN
    IF TYPE.HCC = "UNIX" THEN CALL UNIX
    CALL FW.FREE
  END

RETURN
```

```
/* FW.WAIT ------------------------------------------------------------*/
/* TAKE A NAP BETWEEN CYCLES                                          */
/*--------------------------------------------------------------------*/
FW.WAIT:
  PUSH "FW-I07 FW WILL SLEEP FOR" CYCLE.INTERVAL "SECONDS"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  ADDRESS SYSCALL "SLEEP" CYCLE.INTERVAL
  PUSH "  -    FW IS WAKING UP"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* FW.HOST ------------------------------------------------------------*/
/* NEW HOST PARM FOUND                                                */
/*--------------------------------------------------------------------*/
FW.HOST:
  HC=HC + 1;FC=0
  PUSH FW.PARM
  PARSE PULL PN1 '=' PV1 ',' PN2 '=' PV2 ',' PN3 '=' PV3
  PN1=STRIP(PN1);PN1=PN1'.'HC;PV1=STRIP(PV1);PN1=VALUE(PN1,PV1)
  PN2=STRIP(PN2);PN2=PN2'.'HC;PV2=STRIP(PV2);PN2=VALUE(PN2,PV2)
  PN3=STRIP(PN3);PN3=PN3'.'HC;PV3=STRIP(PV3);PN3=VALUE(PN3,PV3)
  PUSH "FW-I04 PROCESSING PARAMETERS FOR HOST" HOST.HC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* FW.FILE ------------------------------------------------------------*/
/* NEW FILE PARM FOUND                                                */
/*--------------------------------------------------------------------*/
FW.FILE:
  FC = FC + 1
  PUSH FW.PARM
  PARSE PULL PARMNAME '=' PARMVALU
  PARMNAME=STRIP(PARMNAME)
  PARMNAME=PARMNAME'.'HC'.'FC
  PARMVALU=STRIP(PARMVALU)
  PARMNAME=VALUE(PARMNAME,PARMVALU)
  PUSH "  -    FILE =" FILE.HC.FC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN
```

```
/* FW.RENM ------------------------------------------------------------*/
/* NEW RENAME PARM FOUND                                              */
/*--------------------------------------------------------------------*/
FW.RENM:
  PUSH FW.PARM
  PARSE PULL PARMNAME '=' PARMVALU
  PARMNAME=STRIP(PARMNAME)
  PARMNAME=PARMNAME'.'HC'.'FC
  PARMVALU=STRIP(PARMVALU)
  PARMNAME=VALUE(PARMNAME,PARMVALU)
  PUSH "  -      RENAME =" RENAME.HC.FC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* FW.CHKG ------------------------------------------------------------*/
/* NEW CHECKGROWTH PARM FOUND                                         */
/*--------------------------------------------------------------------*/
FW.CHKG:
  PUSH FW.PARM
  PARSE PULL PARMNAME '=' PARMVALU ',' CGSECONDS
  PARMNAME=STRIP(PARMNAME)
  PARMNAME=PARMNAME'.'HC'.'FC
  PARMVALU=STRIP(PARMVALU)
  PARMNAME=VALUE(PARMNAME,PARMVALU)
  PARMVALU=STRIP(CGSECONDS)
  CGSECONDS.HC.FC = CGSECONDS
  PUSH "  -      CHECKGROWTH =" CHECKGROWTH.HC.FC CGSECONDS
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* FW.ACTN ------------------------------------------------------------*/
/* NEW ACTION PARM FOUND                                              */
/*--------------------------------------------------------------------*/
FW.ACTN:
  PUSH FW.PARM
  PARSE PULL PARMNAME '=' PARMTYPE '=' PARMVALU
  PARMNAME=STRIP(PARMNAME)
  PARMTYPE=STRIP(PARMTYPE)
  PARMVALU=STRIP(PARMVALU)
  PARMNAME=PARMNAME'.'HC'.'FC
  PARMNAME=VALUE(PARMNAME,PARMTYPE)
  ACTTYPE.HC.FC = PARMVALU
  PUSH "  -      ACTION =" ACTION.HC.FC "=" ACTTYPE.HC.FC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN
```

```
/* FW.HEND ------------------------------------------------------------*/
/* NEW HOST END PARM FOUND                                            */
/*--------------------------------------------------------------------*/
FW.HEND:
  FI.HC=FC
RETURN

/* FW.ALOC ------------------------------------------------------------*/
/* ALLOCATE LOGON DD                                                  */
/*--------------------------------------------------------------------*/
FW.ALOC:
  "ALLOC DA('"LOGON.PDS"("LOGON.HCC")') DD(LOGON) SHR"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR CANNOT ALLOCATE LOGON DD"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
RETURN

/* FW.WRTE.SYSIN ------------------------------------------------------*/
/* WRITE UCMD PARAMETERS TO SYSIN DD                                  */
/*--------------------------------------------------------------------*/
FW.WRTE.SYSIN:
  PUSH "-i" HOST.HCC "-G YES -x LOGON -s SCRIPT -l" UCMD.TRACE
  "EXECIO 1 DISKW SYSIN (FINIS"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR CANNOT WRITE TO SYSIN DD"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
RETURN

/* UNIX ---------------------------------------------------------------*/
/* PROCESS UNIX FILES                                                 */
/*--------------------------------------------------------------------*/
```

```
UNIX:
  PUSH "FW-I06 CONNECTING TO" HOST.HCC "TO LOOK FOR FILES"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  DO FCC = 1 TO FI.HCC
    FOUND.HCC.FCC = "NO"
    PUSH "ls 2>&1" FILE.HCC.FCC
    "EXECIO" 1 "DISKW SCRIPT"
  END
  "EXECIO" 0 "DISKW SCRIPT (FINIS"

  "CALL '"sysdsname"(UCMD)'"
  IF RC > 2 THEN DO
    IF RC = '1001' THEN DO
      PUSH "FW-XXX ERROR CANNOT CONNECT TO SERVER" HOST.HCC "SEE SYSOUT DD"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      "SEND '"MSG.PREFIX" CANNOT CONNECT TO SERVER" HOST.HCC "SEE SYSOUT DD'"
      RETURN
    END
    FW.CC=12
    PUSH "FW-XXX ERROR UCMD RC NOT = 0 CHECK SYSOUT DD MESSAGES"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END

  "EXECIO * DISKR UNVOUT (FINIS"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR READING FROM UNVOUT"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
  IF FI.HCC <> QUEUED() THEN DO
    FW.CC=12
    PUSH "FW-XXX DIDN'T GET EXPECTED OUTPUT FROM UNVOUT"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
 DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      SIGNAL FW.EXIT
    END
  END
```

```
  DO FCC = 1 TO FI.HCC
    PARSE PULL FW.UNVOUT
    IF SUBSTR(FW.UNVOUT,1,4) = " ls:" THEN ITERATE
    FOUND.HCC.FCC = "YES"
  END

 DO FCC = 1 TO FI.HCC
    IF FOUND.HCC.FCC = "YES" THEN DO
      PUSH "  -    FILE" FILE.HCC.FCC "EXISTS ... PROCESSING"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      CALL UNIX.FOUNDFILE
    END
  END
RETURN

 /* WIN ---------------------------------------------------------------*/
/* PROCESS WINDOWS FILES                                               */
/*-------------------------------------------------------------------*/
WIN:
  PUSH "FW-I06 CONNECTING TO" HOST.HCC "TO LOOK FOR FILES"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  PUSH "@ECHO OFF"
  "EXECIO 1 DISKW SCRIPT"
  DO FCC = 1 TO FI.HCC
    FOUND.HCC.FCC = "NO"
    PUSH "DIR /B 2>&1" FILE.HCC.FCC
    "EXECIO" 1 "DISKW SCRIPT"
  END

  "EXECIO" 0 "DISKW SCRIPT (FINIS"
 "CALL '"sysdsname"(UCMD)'"
  IF RC > 1 THEN DO
    IF RC = '1001' THEN DO
      PUSH "FW-XXX ERROR CANNOT CONNECT TO SERVER" HOST.HCC "SEE SYSOUT DD"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      "SEND '"MSG.PREFIX" CANNOT CONNECT TO SERVER" HOST.HCC "SEE SYSOUT DD'"
      RETURN
    END
    FW.CC=12
    PUSH "FW-XXX ERROR UCMD RC NOT = 0 CHECK SYSOUT DD MESSAGES"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
```

```
 "EXECIO * DISKR UNVOUT (FINIS"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR READING FROM UNVOUT"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
  IF FI.HCC <> QUEUED() THEN DO
    FW.CC=12
    PUSH "FW-XXX DIDN'T GET EXPECTED OUTPUT FROM UNVOUT"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      SIGNAL FW.EXIT
    END
  END
 DO FCC = 1 TO FI.HCC
    PARSE PULL FW.UNVOUT
    IF FW.UNVOUT = " File Not Found" THEN ITERATE
    FOUND.HCC.FCC = "YES"
  END

   DO FCC = 1 TO FI.HCC
    IF FOUND.HCC.FCC = "YES" THEN DO
      PUSH "  -    FILE" FILE.HCC.FCC "EXISTS ... PROCESSING"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      CALL WIN.FOUNDFILE
    END
  END
RETURN

/* UNIX.FOUNDFILE ----------------------------------------------------*/
/* WE FOUND A UNIX FILE LETS PROCESS IT                               */
/*-------------------------------------------------------------------*/

 UNIX.FOUNDFILE:

  IF CHECKGROWTH.HCC.FCC = "NO" THEN NOP
  ELSE CALL UNIX.CHECKGROWTH
  IF GROWING = "YES" THEN RETURN

  IF RENAME.HCC.FCC = "NO" THEN NOP
  ELSE CALL UNIX.RENAME
```

```
  IF ACTION.HCC.FCC = "JCL" THEN CALL FW.JCL
  IF ACTION.HCC.FCC = "MSG" THEN CALL FW.MSG
  IF ACTION.HCC.FCC = "TSO" THEN CALL FW.TSO

RETURN

/* WIN.FOUNDFILE ------------------------------------------------------*/
/* WE FOUND A WINDOWS FILE LETS PROCESS IT                             */
/*--------------------------------------------------------------------*/
WIN.FOUNDFILE:

  IF CHECKGROWTH.HCC.FCC = "NO" THEN NOP
  ELSE CALL WIN.CHECKGROWTH
  IF GROWING = "YES" THEN RETURN

  IF RENAME.HCC.FCC = "NO" THEN NOP
  ELSE CALL WIN.RENAME

  IF ACTION.HCC.FCC = "JCL" THEN CALL FW.JCL
  IF ACTION.HCC.FCC = "MSG" THEN CALL FW.MSG
  IF ACTION.HCC.FCC = "TSO" THEN CALL FW.TSO

RETURN

/* FW.JCL -------------------------------------------------------------*/
/* ACTION = JCL LETS SUBMIT THE JOB                                    */
/*--------------------------------------------------------------------*/
FW.JCL:
  PUSH "  -      JCL REQUESTED .." ACTTYPE.HCC.FCC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  x = OUTTRAP('FWSUBMIT.')
  "SUBMIT '"ACTTYPE.HCC.FCC"'"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO SUBMIT JCL"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    SIGNAL FW.EXIT
  END
  Y = OUTTRAP('OFF')
  PUSH "  -      "FWSUBMIT.1
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN
```

```
 /* FW.TSO ------------------------------------------------------------*/
/* ACTION = TSO LETS ISSUE THE TSO COMMAND                            */
/*-------------------------------------------------------------------*/
FW.TSO:
  PUSH "  -      TSO COMMAND REQUESTED .." ACTTYPE.HCC.FCC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  x = OUTTRAP('FWTSOCMD.')
  ACTTYPE.HCC.FCC
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO ISSUE TSO COMMAND"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    SIGNAL FW.EXIT
  END
 Y = OUTTRAP('OFF')
  DO i = 1 TO FWTSOCMD.0
    PUSH "  -      "FWTSOCMD.i
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
  END
RETURN

/* FW.MSG ------------------------------------------------------------*/
/* ACTION = MSG LETS WRITE THE MESSAGE                                */
/*-------------------------------------------------------------------*/
FW.MSG:
  PUSH "  -      MSG REQUESTED .."
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  "SEND '"MSG.PREFIX ACTTYPE.HCC.FCC"' CN(0)"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO SEND MESSAGE"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    SIGNAL FW.EXIT
  END
  PUSH "  -      MESSAGE SENT TO CONSOLE"
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN
 /* WIN.CHECKGROWTH ---------------------------------------------------*/
/* CHECKING THE GROWTH OF THE FILE WE FOUND - IF REQUESTED            */
/*-------------------------------------------------------------------*/
```

```
WIN.CHECKGROWTH:
  PUSH "  -      GROWTH CHECK REQUESTED .."
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  SAVESIZE.HCC.FCC = "."
  GROWING = "NO"
  PUSH "DIR /N 2>&1" FILE.HCC.FCC
  PUSH "@ECHO OFF"
  "EXECIO 2 DISKW SCRIPT (FINIS"
  WIN.CHECKGROWTH.AGAIN:
  "CALL '"sysdsname"(UCMD)'"

 IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO CHECK GROWTH"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END
    SIGNAL FW.EXIT
  END
  "EXECIO * DISKR UNVOUT (FINIS"
  DO WHILE QUEUED() > 0
    PARSE PULL FW.UNVOUT
    IF SUBSTR(FW.UNVOUT,4,1) = "/" THEN NOP
    ELSE ITERATE
    SIZE.HCC.FCC = SUBWORD(FW.UNVOUT,3,1)
    IF SAVESIZE.HCC.FCC = "." THEN DO
      SAVESIZE.HCC.FCC = SIZE.HCC.FCC
      PUSH "  -      SLEEPING FOR" CGSECONDS.HCC.FCC "SECONDS"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      ADDRESS SYSCALL "SLEEP" CGSECONDS.HCC.FCC
      PUSH "  -      FW IS WAKING UP"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      SIGNAL WIN.CHECKGROWTH.AGAIN
    END
    IF SAVESIZE.HCC.FCC = SIZE.HCC.FCC THEN DO
      PUSH "  -      FILE IS COMPLETE"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END
```

```
 ELSE DO
      PUSH "  -      FILE IS GROWING WILL RECHECK NEXT CYCLE"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      GROWING = "YES"
    END
  END
RETURN

/* UNIX.CHECKGROWTH ------------------------------------------------*/
/* CHECKING THE GROWTH OF THE FILE WE FOUND - IF REQUESTED          */
/*-----------------------------------------------------------------*/
UNIX.CHECKGROWTH:
  PUSH "  -      GROWTH CHECK REQUESTED .."
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  SAVESIZE.HCC.FCC = "."
  GROWING = "NO"
  PUSH "ls -l 2>&1" FILE.HCC.FCC
  "EXECIO 1 DISKW SCRIPT (FINIS"
  UNIX.CHECKGROWTH.AGAIN:
  "CALL '"sysdsname"(UCMD)'"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO CHECK GROWTH"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END
    SIGNAL FW.EXIT
  END
 "EXECIO * DISKR UNVOUT (FINIS"
  DO WHILE QUEUED() > 0
    PARSE PULL FW.UNVOUT
    SIZE.HCC.FCC = SUBWORD(FW.UNVOUT,5,1)
    IF SAVESIZE.HCC.FCC = "." THEN DO
      SAVESIZE.HCC.FCC = SIZE.HCC.FCC
      PUSH "  -      SLEEPING FOR" CGSECONDS.HCC.FCC "SECONDS"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      ADDRESS SYSCALL "SLEEP" CGSECONDS.HCC.FCC
      PUSH "  -      FW IS WAKING UP"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      SIGNAL UNIX.CHECKGROWTH.AGAIN
    END
```

```
    IF SAVESIZE.HCC.FCC = SIZE.HCC.FCC THEN DO
      PUSH "  -      FILE IS COMPLETE"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END
 ELSE DO
      PUSH "  -      FILE IS GROWING WILL RECHECK NEXT CYCLE"
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
      GROWING = "YES"
    END
  END
RETURN

/* WIN.RENAME ------------------------------------------------------*/
/* RENAME THE FILE WE JUST FOUND - IF REQUESTED                     */
/*------------------------------------------------------------------*/
WIN.RENAME:
  PUSH "  -      FILE RENAME REQUESTED .."
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  PUSH "MOVE /Y 2>&1" FILE.HCC.FCC RENAME.HCC.FCC
  PUSH "@ECHO OFF"
  "EXECIO 2 DISKW SCRIPT (FINIS"
  "CALL '"sysdsname"(UCMD)'"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO RENAME FILE"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END

    SIGNAL FW.EXIT
  END
 PUSH "  -      FILE RENAMED TO" RENAME.HCC.FCC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* UNIX.RENAME -----------------------------------------------------*/
/* RENAME THE FILE WE JUST FOUND - IF REQUESTED                     */
/*------------------------------------------------------------------*/
```

```
          UNIX.RENAME:
  PUSH "  -      FILE RENAME REQUESTED .."
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
  PUSH "mv 2>&1" FILE.HCC.FCC RENAME.HCC.FCC
  "EXECIO 1 DISKW SCRIPT (FINIS"
  "CALL '"sysdsname"(UCMD)'"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR TRYING TO RENAME FILE"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    "EXECIO * DISKR UNVOUT (FINIS"
    DO WHILE QUEUED() > 0
      PARSE PULL FW.UNVOUT
      PUSH FW.UNVOUT
      "EXECIO 1 DISKW FWLOGFDD (FINIS"
    END
    SIGNAL FW.EXIT
  END
  PUSH "  -      FILE RENAMED TO" RENAME.HCC.FCC
  "EXECIO 1 DISKW FWLOGFDD (FINIS"
RETURN

/* FW.FREE.SYSIN-------------------------------------------------------*/
/* FREE SYSIN AND LOGON DD                                            */
/*--------------------------------------------------------------------*/
FW.FREE:
  "FREE DD(LOGON)"
  IF RC <> 0 THEN DO
    FW.CC=12
    PUSH "FW-XXX ERROR CANNOT FREE LOGON DD"
    "EXECIO 1 DISKW FWLOGFDD (FINIS"
    SIGNAL FW.EXIT
  END
RETURN
```

Figure 13.16 REXX Script - Watch for One or More Files and Perform Action

## 13.5.4 JCL

Figure 13.17, below, illustrates how to execute the Figure 13.16 script from a z/OS system.

```
 //STEP1    EXEC PGM=IKJEFT01,PARM='USCRP007'
//* TSOBATCH DATASETS
//SYSEXEC  DD  DISP=SHR,DSN=#FWHLQ.FW.CNTL
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//* UNIVERSAL COMMAND DATASETS
//UNVLOAD  DD  DISP=SHR,DSN=#UCHLQ.UNV.SUNVLOAD
//UNVNLS   DD  DISP=SHR,DSN=#UCHLQ.UNV.SUNVNLS
//UNVIN    DD  DUMMY
//UNVERR   DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//* FW DATASETS
//FWCONFDD DD  DISP=SHR,DSN=#FWHLQ.FW.CNTL(FWCONFDD)
//FWPARMDD DD  DISP=SHR,DSN=#FWHLQ.FW.CNTL(FWPARMDD)
//FWLOGFDD DD  SYSOUT=*
//* TEMP DATASETS
//UNVOUT    DD DISP=(NEW,DELETE),DSN=&&UNVOUT,
//             SPACE=(CYL,(1,1)),UNIT=#UNIT,VOL=SER=#VOLSER,
//             DCB=(RECFM=FBA,LRECL=255,BLKSIZE=0)
//SCRIPT    DD DISP=(NEW,DELETE),DSN=&&COMMANDS,
//             SPACE=(CYL,(1,1)),UNIT=#UNIT,VOL=SER=#VOLSER,
//             DCB=(RECFM=FB,LRECL=255,BLKSIZE=0)
//SYSIN     DD DISP=(NEW,DELETE),DSN=&&SYSIN,
//             SPACE=(CYL,(1,1)),UNIT=#UNIT,VOL=SER=#VOLSER,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
```

Figure 13.17 JCL for REXX Script - Watch for One or More Files and Perform Action

The JCL is used to execute the REXX script **USCRP007**. The script will execute a specified command when a specified file is found. It reads member **FWPARMDD** in the data set specified by DD **FWPARMDD** to get a list of files to search for. It reads member **FWCONFDD** specified by DD **FWCONFDD** to get configuration parameters.

When a file is found, the script pauses, based on the **checkgrowth** parameters, in order to confirm that the file is not growing. If the file still is growing, it is skipped until the next cycle. If the file has not grown in this time, the command associated with the file is executed.

## Configuration File

Figure 13.18, below, illustrates a sample configuration file.

```
# FILE WATCHER CONFIGURATION FILE
#

# CYCLE.INTERVAL = seconds
# Specifies the number of seconds to wait between FILE WATCHER cycles.
# DEFAULT is 60 seconds.
#
CYCLE.INTERVAL = 60

# MSG.PREFIX
# Specifies the PREFIX for message requests
# DEFAULT = no prefix, messages are sent as specified in FWPARMDD
#
MSG.PREFIX = FW-MSG

# LOGON.PDS
# Specifies the PDS dataset that contains the members specified with
# the LOGON parameter.
#
LOGON.PDS = PROD.LOGON.CNTL

# UCMD.TRACE
# Sets message level for UCMD calls made within FW.
# Specify a valid message level, default = ERROR
#
UCMD.TRACE = ERROR

# REXX.TRACE
# Turns on REXX TRACE R. Trace output is written to SYSTSPRT DD.
# Specify YES or NO, Default = NO
#
REXX.TRACE = NO
```

Figure 13.18 Configuration File

## Parameter File

The following example shows a sample parameter file used to specify file and host parameters.

```
# FILE WATCHER PARAMETER FILE used with script USCRP007
#
# FORMAT EXAMPLE

#
# PARAMETERS FOR AIX51
#
HOST=UNIXHOST,LOGON=USERMEMBER,TYPE=UNIX

FILE=/home/test1
RENAME=/home/test2
CHECKGROWTH=YES,10
ACTION=JCL=PROD.JCL.CNTL(IEFBR14)

FILE=/usr/local/test2
RENAME=NO
CHECKGROWTH=NO
ACTION=MSG=THIS FILE HAS BEEN FOUND

END HOST
#
# PARAMETERS FOR NT SERVER
#
HOST=NTHOST,LOGON=NTUSER,TYPE=WIN

FILE=c:\temp\test.txt
RENAME=NO
CHECKGROWTH=NO
ACTION=JCL=PROD.JCL.CNTL(IEFBR14)

FILE=c:\"program files\universal\test.txt"
RENAME=NO
CHECKGROWTH=NO
ACTION=TSO=TIME

FILE=c:\temp\test3.txt
RENAME=NO
CHECKGROWTH=NO
ACTION=MSG=TEST3.TXT HAS BEEN FOUND

END HOST
```

Figure 13.19 Parameter File

# 13.6 Remote Editing

Figure 13.20, below, illustrates a script that can be executed to access a file on a remote UNIX system and edit that file from a local Windows NT/2000.

Universal Command is used to copy the requested file from a remote UNIX system to a temporary file on the local WIndows system.

Next, the temporary file is opened in the Notepad editor. Notepad does not have to be used to edit the file. You can update the **`edx.cmd`** file to invoke any NT-based text file editor that you want.

After you exit the text editor, Universal Command is invoked to copy the file back to the original system. All temporary files then are removed.

```
@echo off
rem ************************************************************************
rem * EDX.CMD - edit files on other systems.                               *
rem * cmdfile - edxcmd.txt – universal command parameters pointing to      *
rem *           remote UNIX server.                                        *
rem * filename – remote file to edit locally.                              *
rem *                                                                      *
rem * Syntax:                                                              *
rem * EDX cmdfile filename                                                 *
rem *                                                                      *
rem * Example:                                                             *
rem * EDX edxcmd.txt /etc/hosts                                           *
rem *                                                                      *
rem ************************************************************************
ucmd -f %1 -c "cat %2" 1> edxtemp.txt 2>edx1.log
notepad edxtemp.txt
ucmd -f %1 -c "cat > %2" < edxtemp.txt 2>edx2.log
type edx1.log
type edx2.log
del edx1.log
del edx2.log
del edxtemp.txt
```

Figure 13.20 Remote Editing Script

## 13.6.1 Command File

The following example shows how to code the parameter file **edxcmd.txt**. This should be customized to point to the UNIX host where the remote file resides.

```
-host hostname -u userid -w password
```

## 13.6.2 Example

The following example shows how to execute the above script from an NT server command line. The example will edit the **hosts** file in the **/etc** directory of a remote UNIX system.

```
EDX edxcmd.txt /etc/hosts
```

### SYSIN Options

The SYSIN options used are:

| SYSIN Options | Description |
|---|---|
| -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. |

# Appendix A Examples

## A.1 Overview

This appendix provides operating system-specific examples that demonstrate the use of Universal Command:

- UCMD Manager for z/OS Examples
- UCMD Manager for Windows Examples
- UCMD Manager for UNIX Examples
- UCMD Manager for OS/400 Examples
- UCMD Manager for HP NonStop Examples

# A.2 UCMD Manager for z/OS Examples

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

Note: 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:

- Directory Listing for UNIX Server
- Directory Listing for Windows Server
- Netstat Command for Windows
- Netstat Command for UNIX
- z/OS to UNIX Backup to z/OS Dataset
- z/OS to UNIX Backup Restore from z/OS Dataset
- UNIX Copy to z/OS Dataset
- Script Commands to Windows
- JCL Procedure
- Manager Command to Windows, UNIX, OS/400, HP NonStop
- Command Coded as a Script
- Override Standard Files with DDNAME
- Override Standard Files with SYMBOLICS
- Override Command Line Parameters from the Execute Statement
- Authentication Parameters from Encrypted File
- Override Restart Parameter in Command Line
- Override Restart Parameter in PARM
- Redirect Standard Out to File and UCMD Manager
- Execute a Windows .bat file
- Unique Command ID with CA-7 or CA-Scheduler
- Unique Command ID with Zeke
- Unique Command ID with OPC

# A.2.1 Directory Listing for UNIX Server

```
//UNIXLS JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//COMMANDS DD  *
 ls -l /opt/universal/bin       <-----This is the UNIX command.
//SYSIN    DD  *
 -host               Unix_1
 -userid             username
 -pwd                password
 -script             COMMANDS
/*
```

Figure A.1 UCMD Manager for z/OS – Directory Listing for UNIX Server

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **176.16.30.20**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.2 Directory Listing for Windows Server

```
//WINDIR  JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//COMMANDS DD  *
  cd\             <------ Change to the desired directory.
  dir             <------ Directory information will be in JES2.
output
//SYSIN    DD  *
  -host      Window_1
  -userid    userid
  -pwd       password
  -script    COMMANDS
/*
```

Figure A.2 UCMD Manager for z/OS – Directory Listing for Windows Server

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **176.16.30.30**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.3 Netstat Command for Windows

```
//WSTATUS JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//COMMANDS DD  *
  netstat -r
//SYSIN    DD  *
 -host                Window_1
 -userid              username
 -pwd                 Liimotb1
 -script              COMMANDS
/*
//* There are multiple switches to see various displays; -r shows routing
    table.
```

Figure A.3 UCMD Manager for z/OS – Netstat Command for Windows

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Window_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.4 Netstat Command for UNIX

```
//USTATUS JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//COMMANDS DD  *
  netstat  -e
//SYSIN    DD  *
 -host           Unix_1
 -userid         username
 -pwd            password
 -script         COMMANDS
/*
//* There are multiple switches to see various displays; -e displays Ethernet
    stats.
```

Figure A.4 UCMD Manager for z/OS – Netstat Command for UNIX

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Unix_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.5 z/OS to UNIX Backup to z/OS Dataset

```
//UBACKUP  JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*****************************************************************
//* Description
//* -----------
//* This sample backs up the UNIX directory /export/home/username/fnd
//* to a local preallocated dataset.
//*
//         JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,
//         STDOUT='DISP=SHR,DSN=username.BCUP.TAR'
//*
//MYSCRIPT DD  *
cd /export/home/username/fnd
tar -cvf - . | compress
//SYSIN    DD  *
   -host Unix_1
   -userid  username
   -pwd  password
   -script MYSCRIPT
   -stdout -mode binary
/*
```

Figure A.5 UCMD Manager for z/OS – z/OS to UNIX Backup to z/OS Dataset

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Unix_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |
| -stdout | Starts the stdout options. All options read afterwards are applied to the stdout file. The first option not recognized as a standard file option terminates the stdout option list. |
| -mode | Specifies a transfer mode of **binary** for the stdout file. The data is not translated. |

# A.2.6 z/OS to UNIX Backup Restore from z/OS Dataset

```
//URESTORE JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*****************************************************************
//* Description
//* -----------
//* This sample restores the UNIX directory /export/home/username/fnd
//*****************************************************************
//*
//         JCLLIB ORDER=username.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,
//         STDIN='DISP=SHR,DSN=username.BCUP.TAR'
//MYSCRIPT DD  *
if test ! -d /export/home/username/fnd
  then
    mkdir /export/home/username/fnd
  fi
#
# Note: The above test looks for the directory
#       if there is not directory it builds it
#
cd /export/home/username/fnd

# Note: Not all tar commands recognize the 'B' argument.  If you
#       receive an error message indicating this from the remote
#       UNIX system, remove the 'B' argument.
#       The 'B' argument is used to force tar to read multiple
#       times to fill the block.
uncompress | tar -xvBf -
diff . /export/home/username/fnd
//SYSIN    DD  *
 -script myscript -host Unix_1  -userid username  -pwd password
 -stdin -mode binary
/*
```

Figure A.6 UCMD Manager for z/OS – z/OS to UNIX Backup Restore from z/OS Dataset

## 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 host address **`Unix_1`**. |
| `-userid` | Specifies the remote user ID with which to execute the command. |
| `-pwd` | Specifies the password for the user ID. |
| `-stdin` | Starts the stdin options. All options read afterwards are applied to the stdin file. The first option not recognized as a standard file option terminates the stdin option list. |
| `-mode` | Specifies a transfer mode of **`binary`** for the stdin file. The data is not translated. |

# A.2.7 UNIX Copy to z/OS Dataset

```
//UNIXCAT  JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//         JCLLIB ORDER=username.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,
//               STDOUT='DISP=SHR,DSN=username.UNIX.FILE1'
//SYSIN    DD  *
 -cmd 'cat /export/home/username/file1'
     -host Unix_1 -userid username -pwd password -l audit
/*
```

Figure A.7 UCMD Manager for z/OS – UNIX Copy to z/OS Dataset

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **cat /export/home/username/file1** to execute. The **cat** program copies the files specified on the command line to its **stdout.n**. |
| -host | Directs the command to a computer with host address **Unix_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -l | Specifies the message level output for this command execution. |

# A.2.8 Script Commands to Windows

```
//WINSCRI JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//SCRIPTDD DD  DISP=SHR,DSN=username.JCL.CNTRL(WSCRIPT)
//SYSIN    DD  *
 -host                Window_1
 -userid              username
 -pwd                 password
 -script              SCRIPTDD
/*
```

Figure A.8 UCMD Manager for z/OS – Script Commands to Windows

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Window_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.9 JCL Procedure

Figure A.9, below, illustrates a JCL procedure in order to simplify the execution JCL and future maintenance.

Multiple version of the procedure can be created in order to point to different default configuration members. The procedure provides the symbolic STDIN, STDOUT, and STDERR to specify the standard in, standard out, and standard error files, respectively, of the remote command. DD Names of **UNVIN**, **UNVOUT**, and **UNVERR** point to the standard in, standard out, and standard error files. The parameter **UPARM** is used to specify EXEC PARM keyword values. The parameter **CONFIG** specifies the configuration member. Lastly, parameter **UCMDPRE** specifies the data set name prefix of Universal Products installation data sets.

## HLQ.UNV.SUNVSAMP

```
//UCMDPRC  PROC UPARM=,
//             STDOUT='SYSOUT=*',
//             STDERR='SYSOUT=*',
//              ='DUMMY',
//             CONFIG=UCMCFG00,
//             UCMDPRE=UNV
//*
//PS1      EXEC PGM=UCMD,PARM='&UPARM'
//STEPLIB  DD  DISP=SHR,DSN=&UCMDPRE..SUNVLOAD
//*
//UNVNLS   DD  DISP=SHR,DSN=&UCMDPRE..SUNVNLS
//UNVCONF  DD  DISP=SHR,DSN=&UCMDPRE..UNVCONF(&CONFIG)
//UNVTRACE DD  SYSOUT=*
//*
//UNVOUT   DD  &STDOUT
//UNVERR   DD  &STDERR
//UNVIN    DD  &STDIN
//*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
```

Figure A.9 UCMD Manager for z/OS - JCL Procedure

# A.2.10 Manager Command to Windows, UNIX, OS/400, HP NonStop

Figure A.10, below, illustrates the minimum required parameters for the UCMD Manager, the command to be executed, and the remote server on which to execute. Security is set to default on the remote server so authentication parameters of user ID and password also are required.

```
//stepname EXEC UCMDPRC
//SYSIN DD *
-cmd 'dir \' -host dallas
-userid joe –pwd abcdefg
/*
```

Figure A.10 UCMD Manager for z/OS - Manager Command to Windows, UNIX, OS/400, 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 of the **DIR** command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to SYSOUT in the **UCMDPRC** procedure. The process will authenticate and run under the authority of user id **joe**.

If the remote system is a UNIX system, change the command **dir ** to **ls /**. If the remote system is OS400, change the command **dir ** to **DSPLIB QGPL**. If the remote system is TACL HP NonStop, change the command **dir ** to **files**. If the remote system is OSS HP NonStop, change the command **dir ** to **ls**.

## SYSIN Options

SYSIN options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command '**dir **' to execute. |
| -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. |

# A.2.11 Command Coded as a Script

Figure A.11, below, illustrates how to code the remote command as a script.

The benefits are:

- Multiple commands can be executed with one UCMD Manager.
- The script can be housed separate from the execution JCL.
- The script can be stored and promoted via change control procedures.
- Provides separation of the remote command from UCMD SYSIN.

```
//UNISCRI JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//SCRIPTDD DD  DISP=SHR,DSN=username.JCL.CNTRL(USCRIPT)
//SYSIN    DD  *
 -host              Unix_1
 -userid            username
 -pwd               password
 -script            SCRIPTDD
/*
//
```

Figure A.11 UCMD Manager for z/OS - Command Coded as a Script

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Unix_1** for execution. The process will authenticate and run under the authority of user id **username**.

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Unix_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.12 Override Standard Files with DDNAME

Figure A.12, below, illustrates how to override the standard input, output, and error using the DDNAMES coded in the UCMD Procedure. The DDNAMES can be overridden either to point to data sets instead of **SYSOUT** or to change the **SYSOUT** class.

```
//UNISCRD JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  DISP=SHR,DSN=username.JCL.CNTRL(OUTPUT)
//* UNIVERSAL COMMAND WILL CREATE THE MEMBER
//UNVERR   DD  SYSOUT=*
//SCRIPTDD DD  DISP=SHR,DSN=username.JCL.CNTRL(USCRIPT)
//SYSIN    DD  *
 -host               Unix_1
 -userid             username
 -pwd                password
 -script             SCRIPTDD
/*
```

Figure A.12 UCMD Manager for z/OS - Override Standard Files with DDNAME

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Unix_1** for execution. The process will authenticate and run under the authority of userid **username**.

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Unix_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.13 Override Standard Files with SYMBOLICS

Figure A.13, below, illustrates shows how to override the standard input, output, and error using the SYMBOLICS coded in the Universal Command Procedure.

```
//SYMBOL JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,STDIN='DUMMY',STDERR='SYSOUT=*',
//         STDOUT='DISP=SHR,DSN=username.JCL.CNTRL(OUT2)'
//SCRIPTD  DD DISP=SHR,DSN=username.JCL.CNTRL(WSCRIPT)
//SYSIN    DD  *
 -host                Window_1
 -userid              username
 -pwd                 password
 -script              SCRIPTD
/*
```

Figure A.13 UCMD for z/OS - Override Standard Files with SYMBOLICS

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution. The process will authenticate and run under the authority of userid **username**.

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Window_1**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.14 Override Command Line Parameters from the Execute Statement

Figure A.14, below, illustrates how to override command line parameters from the execute statement using the symbolic PARM coded in the UUCMD Procedure.

```
//PARMLINE JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//         JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,PARM='-restart yes'
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(WINUSER)
//SCRIPTDD DD *
dir \
//SYSIN    DD  *
-script SCRIPTDD -host Window_1 -x LOGONDD -managerft yes
 -cmdid PARMLINE
/*
//* Note that the 'restart yes' cannot be used unless a job needs to
//* restart it will error with  ReceiveRestartResponse, 5, no component
//* with the specified command ID was found
//* Note also that Command IDs cannot be generated for MFT restarts
//* you must supply a unique file name. In this example the job name
//* was used.
```

Figure A.14 UCMD Manager for z/OS - Override Command Line Parameters from Execute Statement

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution.

## 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 host address **Window_1**. |
| -x | Specifies the DD from which to read an encrypted command options file. |
| -managerft | Specifies whether or not the manager fault tolerant feature is used. |
| -cmdid | Specifies an identifier used to identify the unit of work represented by the Manager, Server, and User Command. |

# A.2.15 Authentication Parameters from Encrypted File

Figure A.15, below, illustrates how to read UCMD parameters from an encrypted file. The encrypted file was previously created using the Universal Encrypt utility with the parameters **–userid** and **–pwd**.

```
//ENCRYPT JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//        JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(WINUSER)
//SCRIPTDD DD  DISP=SHR,DSN=username.JCL.CNTRL(WSCRIPT)
//SYSIN    DD  *
 -host Window_1 -x LOGONDD
 -script              SCRIPTDD
/*
```

Figure A.15 UCMD Manager for z/OS - Authentication Parameters from Encrypted File

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution.

## SYSIN Options

The SYSIN options used in this example are:

| Option | Description |
|---|---|
| -host | Directs the command to a computer with host address **Window_1**. |
| -x | Specifies the DD from which to read an encrypted command options file. |
| -script | Specifies the DD from which to read a script file. The script file is sent to the remote system for execution. |

# A.2.16 Override Restart Parameter in Command Line

Figure A.16, below, illustrates how to override the restart parameter in the command line.

```
//CMDLINE  JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//         JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(WINUSER)
//SCRIPTDD DD *
dir \
//SYSIN    DD  *
-script SCRIPTDD -host Window_1 -x LOGONDD -managerft yes
 -cmdid CMDLINE -restart yes
/*
//* Note that the 'restart yes' cannot be used unless a job needs to
//* restart, it will error with  ReceiveRestartResponse, 5, no component
//* with the specified command ID was found
//* Note also that Command IDs cannot be generated for MFT restarts
//* you must supply a unique file name. In this example the job name
//* was used.
```

Figure A.16 UCMD Manager for z/OS - Override Restart Parameter in Command Line

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution. The output of the **dir ** command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to **SYSOUT** in the **UCMDPRC** procedure.

Additional command line options are read from the encrypted file allocated to DD **LOGONDD**. Manager fault tolerance is turned on. A unique command ID is coded to identify the process. Restart is automatically detected. If an executing or pending command ID exists, a reconnect is performed. If not, the process is started as if new.

## 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 host address of **`Window_1`**. |
| `-x` | Specifies the DD from which to read an encrypted command options file. |
| `-managerft` | Specifies if the manager fault tolerant feature is used. |
| `-cmdid` | Unique command ID associated with the unit of work. |
| `-restart` | Specifies whether or not the manager is requesting restart. |

# A.2.17 Override Restart Parameter in PARM

Figure A.17, below, illustrates how to override the restart parameter in the PARM. If your default is set to *no*, then each restart must override the value to *yes*. This can be done through the symbolic PARM.

```
//PARMLINE JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//                    JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC,PARM='-restart yes'
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(WINUSER)
//SCRIPTDD DD *
dir \
//SYSIN    DD  *
-script SCRIPTDD -host Window_1 -x LOGONDD -managerft yes
 -cmdid PARMLINE
/*
//* Note that the 'restart yes' cannot be used unless a job needs to
//* restart, it will error with  ReceiveRestartResponse, 5, no component
//* with the specified command ID was found
//* Note also that Command IDs cannot be generated for MFT restarts
//* you must supply a unique file name. In this example the job name
//* was used.
```

Figure A.17 UCMD Manager for z/OS - Override Restart Parameter in PARM

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution. The output of the **dir ** command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to **SYSOUT** in the **UCMDPRC** procedure.

Additional command line options are read from the encrypted file allocated to DD **LOGONDD**. Manager fault tolerance is turned on. A unique command ID is coded to identify the process. Restart is set to *yes* with the symbolic PARM.

If the remote system is a UNIX system, change the command **dir ** to **ls /**. If the remote system is OS/400, change the command **dir ** to **DSPLIB QGPL**. If the remote system is TACL HP NonStop, change the command **dir ** to **files**. If the remote system is OSS HP NonStop, change the command **dir ** to **ls**.

## 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 host addresss **`Window_1`**. |
| `-x` | Specifies the DD from which to read an encrypted command options file. |
| `-managerft` | Specifies whether or not the manager fault tolerant feature is used. |
| `-cmdid` | Unique command ID associated the unit of work. |

# A.2.18 Redirect Standard Out to File and UCMD Manager

Some applications redirect standard output to a file on the local server.

Figure A.18, below, illustrates how the UNIX **tee** command can be used to write standard out to the local server file and the manager real-time.

```
//THETEE   JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//         JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(UNXUSER)
//SCRIPTDD DD *
ls | tee teeout.txt
//SYSIN    DD  *
-script SCRIPTDD -HOST Unix_1 -x LOGONDD
/*
```

Figure A.18 UCMD Manager for z/OS - Redirect Standard Out to File and UCMD Manager

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Unix_1** for execution. The output of the **ls** command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to SYSOUT in the **UCMDPRC** procedure. At the same time, the output also is written to the file **teeout.txt** on the local server. Additional command line options are read from the encrypted file allocated to DD **LOGONDD**. The **tee** command is available only on UNIX. There are shareware versions for WIndows that can be found.

## 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 host address **Unix_1**. |
| -x | Specifies the DD from which to read an encrypted command options file. |

# A.2.19 Execute a Windows .bat file

Manager-supplied script files are processed as batch files (extension **.bat**) on Windows operating systems.

Figure A.19, below, illustrates how to execute a **.bat** file from within the manager script file, using the **call** command.

```
//WINDOBAT JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//         JCLLIB ORDER=SBI.UNV.SUNVSAMP
//*
//STEP1    EXEC UCMDPRC
//UNVIN    DD  DUMMY
//UNVOUT   DD  SYSOUT=*
//UNVERR   DD  SYSOUT=*
//LOGONDD  DD  DISP=SHR,DSN=username.JCL.CNTRL(WINUSER)
//SCRIPTDD DD *
call user.bat
//SYSIN    DD  *
-script SCRIPTDD -host Window_1 -x LOGONDD
/*
```

Figure A.19 UCMD Manager for z/OS - Execute a Windows .bat File

The JCL procedure **UCMDPRC** is used to execute the command. The command is sent to remote system **Window_1** for execution. The output of the command is redirected back to the Universal Command batch job and written to ddname **UNVOUT**, which is allocated to **SYSOUT** in the **UCMDPRC** procedure. 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 host address **Window_1**. |
| -x | Specifies the DD from which to read an encrypted command options file. |

# A.2.20 Unique Command ID with CA-7 or CA-Scheduler

CA provides variable functionality via CA-Driver. (Refer to the CA manuals for instructions on variable usage.)

CA-Driver variables specific to CA-7 such as the following can be used to create a unique **cmdid**. (Refer to the CA documentation for a complete list of CA-Driver variables available for CA-7 and CA-Scheduler.)

- &C-L2JN is the CA-7 Job Name
- &C_L27# is the CA-7 Job Number
- &C_L2DOD is the Due Out Date for this Job execution.

One Procedure should be placed in the CA-Driver Procedure library.

Figure A.20, below, illustrates an example of a CA-Driver Procedure using the above system variables to create a unique command ID.

```
//DRVRUCMD    DPROC REMOTEJOBNAME=UCMD
-cmdid '&REMOTEJOBNAME.&C_L2JN.&C_L27#.&C_L2DOD'
```

Figure A.20 UCMD Manager for z/OS - CA-Driver Procedure

Each step that executes UCMD should reference this procedure in order to create the unique UCMD **cmdid** as the first parameter within the UCMD SYSIN DD statement. This procedure defaults the **cmdid** to the values defined by one user variable called **remotejobname** and 3 CA-7 variables.

(See the examples on the following pages.)

## Example 1

Figure A.21, below, illustrates how to call the PROC **DRVRUCMD** from within the UCMD SYSIN DD statement.

The variables set in the **DRVRUCMD** PROC are set to the following for this example:

- &C-L2JN = PRD00001
- &C_L27# = 0030001
- &C_L2DOD 03265
- &REMOTEJOBNAME = UCMD (default value in Driver Procedure **DRVRUCMD**)

```
//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
DIR
//SYSIN      DD *
//CALL       EXEC     PROC=DRVRUCMD,REMOTEJOBNAME=
-s scriptdd –host dallas -x logondd –M yes –restart auto
/*

Expanded Results:

//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
dir
//SYSIN      DD *
-cmdid UCMDPRD000010030001032625
-s scriptdd –host dallas –x logondd –M yes –restart auto
/*
```

Figure A.21 UCMD Manager for z/OS - Call PROC DRVRUCMD from within UCMD SYSIN DD statement

* If the **cmdid** identifier contains spaces, it must be enclosed in either single ( **'** ) or double ( **"** ) quotation marks.

## Example 2

Figure A.22, below, illustrates how to override the variable value for **REMOTEJOBNAME** in the **UCMDSTEP**.

```
//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
DIR
//SYSIN      DD *
//CALL       EXEC     PROC=DRVRUCMD,REMOTEJOBNAME=unixpayrolljob1
-s scriptdd –host dallas -x logondd –M yes –restart auto
/*

Expanded Results:

//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
dir
//SYSIN      DD *
-cmdid unixpayrolljob1PRD000010030001032625
-s scriptdd –host dallas –x logondd –M yes –restart auto
/*
```

Figure A.22 UCMD Manager for z/OS - Override Variable Value for REMOTEJOBNAME in UCMDSTEP

* If the **cmdid** identifier contains spaces, it must be enclosed in either single ( **'** ) or double ( **"** ) quotation marks.

# A.2.21 Unique Command ID with Zeke

Zeke has a set of reserved variables available that get substituted during job submission. The default character **$** is used to identify a Zeke variable within the instream JCL. This default character can be changed during installation. Create a variable whose value is set based on the current schedule date. Use this variable in the UCMD Manager jobs.

(See the ASG Zeke documentation for instructions on variable usage.)

```
//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
DIR
//SYSIN      DD *
-cmdid 'jobname$SCHDATEunixpayrolljob1'     <== Zeke  variable
-s scriptdd -host dallas -x logondd -M yes -restart auto
/*

Expanded Results:

//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
dir
//SYSIN      DD *
-cmdid 'JOBNAME03254unixpayrolljob1'
-s scriptdd -host dallas -x logondd -M yes -restart auto
/*
```

Figure A.23 UCMD Manager for z/OS - Unique Command ID with Zeke

* If the **cmdid** identifier contains spaces, it must be enclosed in either single ( **'** ) or double ( **"** ) quotation marks.

# A.2.22 Unique Command ID with OPC

OPC has a set of reserved variables available that get substituted at job submission.

The feature gets switched on by coding the following JCL statements:

- //*%OPC SCAN <== set substitution on and off by
- //*%OPC NOSCAN <== set substitution off

Any OPC variable found within the instream JCL can be substituted with the current value by OPC. (See the IBM OPC documentation for instructions on variable usage.)

```
//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
DIR
//SYSIN    DD  *
-C payrolljob&CYMD.&CHHMMSSX.          <== OPC variables
-s scriptdd -host dallas -x logondd-M yes -restart no
/*

Expanded Results:

//stepname EXEC UCMDPRC
//LOGONDD  DD DISP=SHR,DSN=hlq.encrypted.file
//SCRIPTDD DD *
dir
//SYSIN    DD  *
-C payrolljob2003061613315614
-s scriptdd -host dallas -x logondd -M yes -restart no
/*
```

Figure A.24 UCMD Manager for z/OS - Unique Command ID with OPC

* If the **cmdid** identifier contains spaces, it must be enclosed in either single ( **'** ) or double ( **"** ) quotation marks.

# A.3 UCMD Manager for Windows Examples

This section contains examples demonstrating the use of UCMD Manager for Windows.

The following list provides a link to each example.

# A.3.1 File Copy Example

This example copies a file from a remote UNIX system to a local file.

Figure A.25, below, illustrates the command. Although the command is shown on two lines, it should be entered as one line at the command prompt.

```
ucmd -cmd "cat ~/file" -host dallas
     -userid joe –pwd password -comment "copy ~/file from dallas" > localfile
```

Figure A.25 UCMD Manager for Windows - File Copy Example

The stdout of the **cat** 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 **localfile**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command - **"cat ~/file"** - to execute. The **cat** 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. |
| -comment | Describes the process executed by Universal Command. |

The file is copied as a text file, since the default transfer mode is **text**.

## A.3.2 System Status Report Example

This example produces a report of the system status of a remote UNIX system. Instead of executing a command on the remote host, a local script file is executed.

Figure A.26, below, illustrates the script file **myscript**.

Note: The commands executed in the script file may or may not require modifications depending on the type of UNIX system on which it executes.

```
echo "System Status as of `date`"
echo "-------------------------------------------------"
netstat
echo "-------------------------------------------------"
df
echo "-------------------------------------------------"
ps –ax
```

Figure A.26 UCMD Manager for Windows - System Status Script File Example

Figure A.27, below, illustrates the command to execute the script file.

```
ucmd -script myscript -host dallas -userid joe –pwd password
```

Figure A.27 UCMD Manager for Windows - System Status Example

The report is written to the standard out of UCMD.

### Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file name of 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

## A.3.3 Directory Backup Example

This example backs up a directory and its subdirectories on a UNIX system to a local data set. Instead of executing a command on the remote host, a local script file is executed.

Figure A.28, below, illustrates the script file.

```
cd /usr/man/man1
tar -cv . | compress
```

Figure A.28 UCMD Manager for Windows - Directory Backup Script File Example

Figure A.29, below, illustrates the command to execute the script file.

```
ucmd –script myscript -host dallas -userid joe -pwd password
     –stdout –mode binary > data.tar
```

Figure A.29 UCMD Manager for Windows - Directory Backup Example

The script file changes its current directory to the directory to backup. The **tar** command creates an archive file containing all files and subdirectories located in the current directory. This archive file is written to **tar**'s standard out, which is piped to the **compress** command. The compress command compresses its input and writes to its standard out. The standard out of the compress command is the same standard out of the script file. The script file's standard out is redirected back to the **ucmd** command running on the local system. The standard out of UCMD is redirected to the local file **data.tar**.

### Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file name of 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -stdout | Starts the stdout option list. All options read afterwards are applied to the stdout file. The first option not recognized as a standard file option terminates the stdout option list. |
| -mode | Specifies a transfer mode of binary for the stdout file. The data is not translated. |

## A.3.4 Directory Restore Example

This example restores the directory that was backed up in the previous example. The file containing the backup is on the local system.

Figure A.30, below, illustrates the script to perform the restore.

```
if test ! -d man1
  then
    mkdir man1
  fi

cd man1
uncompress | tar –xvf -
diff . /usr/man/man1
```

Figure A.30 UCMD Manager for Windows - Directory Restore Script Example

Figure A.31, below, illustrates the command to execute the script file.

```
ucmd -script myscript –host dallas –userid joe -pwd password
     -stdin –mode binary < file.tar
```

Figure A.31 UCMD Manager for Windows - Directory Backup Example

The script file creates directory **man1** in **joe**'s home directory if it does not already exist. It then changes its current directory to **man1**. The **uncompress** command reads from the script's standard in file, which is redirected from UCMD's standard in on the local system.

Notice that UCMD's standard in is redirected from the backup file **file.tar**. The **uncompress** program uncompresses its input and writes it to its standard out, which is piped to the **tar** command. The **tar** command extracts and writes the archive to the current directory. The final command, **diff**, compares the original directory with the new one. The **diff** command returns **0** if no differences are found; otherwise, it returns **1**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| `-script` | Specifies the file name of 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`**. |
| `-userid` | Specifies the remote user ID with which to execute the command. |
| `-pwd` | Specifies the password for the user ID. |
| `-stdin` | Starts the stdin option list. All options read afterwards are applied to the stdin file. The first option not recognized as a standard file option terminates the stdin option list. |
| `-mode` | Specifies a transfer mode of binary for the stdout file. The data is not translated. |

# A.3.5 Manager Command

Figure A.32, below, illustrates the minimum required parameters for the Universal Command Manager, the command to be executed (**-cmd**) and the remote server (**-host**) to execute on. Security is set to default on the remote server so authentication parameters of userid and password also are required.

```
ucmd -cmd 'dir' -host dallas –userid joe –pwd password
```

Figure A.32 UCMD Manager for Windows - Manager Command

The command **dir** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **dir** command are available to the initiating process. The process will authenticate and run under the authority of userid **joe**. If the remote system is a UNIX system, change the command **dir** to **ls /**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ls** to execute. |
| -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. |

# A.3.6 Command Coded as a Script

Figure A.33, below, illustrates how to code the remote command as a script.

The benefits are:

- Multiple commands can be executed with one UCMD Manager.
- Script can be housed separate from the execution command.
- Script can be stored and promoted via change control procedures.
- Provides separation of the remote command from UCMD options.

```
ucmd -script script.file -host dallas -userid joe -pwd password
```

Figure A.33 UCMD Manager for Windows - Command Coded as a Script

The contents of the **`script.file`** are sent to a remote system named **`dallas`** for execution. The output of the script command is redirected back to the UCMD process. The standard output and standard error of the script execution are available to the initiating process. The process will authenticate and run under the authority of userid **`joe`**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| `-script` | Specifies the file 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`**. |
| `-userid` | Specifies the remote user ID with which to execute the command. |
| `-pwd` | Specifies the password for the user ID. |

# A.3.7 Redirect Standard Out and Standard Error

Figure A.34, below, illustrates how to redirect the standard output and error of the 'DIR' command to a file on the initiating system.

```
ucmd -cmd 'dir' -host dallas –userid joe –pwd password > output.file 2>&1
```

Figure A.34 UCMD Manager for Windows - Redirect Standard Out and Standard Error

The command **dir** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **dir** command are written back to the UCMD process and redirected to standard out file **output.file**. The process will authenticate and run under the authority of userid **joe**.

If the remote system is a UNIX system, change the command **dir** to **ls**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ls** to execute. |
| -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. |

# A.3.8 Spawn Background Process with nohup: UNIX

The UCMD Manager job will not end until the remote process ends and all standard files are closed. If a script starts a process that will not end until stopped, the manager process will not end until the process is stopped. In order to start the process and continue processing without waiting for the process to be stopped and the close of the standard files, start the process in the background using the **NOHUP** command and redirect standard out and error to **/dev/null**.

```
ucmd –cmd ‘nohup startprocess –host dallas –userid joe –pwd password
> /dev/null 2>&1’
```

Figure A.35 UCMD Manager for Windows - Spawn Background Process with nohup: UNIX

The command to start a process is issued with the UNIX **nohup** parameter. Any output is written to **/dev/null** which never is saved to disk or memory. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ls** to execute. |
| -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. |

# A.3.9 Redirect Standard Input from Initiating System

The **ucmd** command reads from standard input and writes it to the UCMD Server for the remote command to read as its standard input. The allocation of standard input can be changed with a shell redirection operator. The redirection operators instruct the shell to change the allocation of the standard files. To change the allocation of standard input, use the **<** operator.

```
ucmd –script myscript -host dallas –userid joe –pwd password < input.file
```

Figure A.36 UCMD Manager for Windows - Redirect Standard Input from Initiating System

The command is sent to a remote system named **dallas** for execution. The output of the script is redirected back to the Universal Command process's standard out and standard error. Standard input is read from file **input.file** on the initiating system. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.3.10 Redirect Standard Input from /dev/null

If the command **ucmd** is executed as a background job (using the **&** operator), it will receive the SIGTTIN signal when **ucmd** tries to read from standard input. Background jobs cannot read their standard input from the terminal since the foreground job (or the shell) has it allocated. The **ucmd** job is stopped until it is brought to the foreground. To run an **ucmd** job that does not require terminal input in the background, redirect its standard input from **/dev/null**.

```
ucmd –script &myscript –host dallas –userid joe –pwd password < /dev/null
```

Figure A.37 UCMD Manager for Windows - Redirect Standard Input from /dev/null

The command is sent to a remote system named **dallas** for execution. The output of **myscript** is redirected back to the Universal Command process. Standard input is read from **/dev/null**. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.3.11 Authentication Parameters from Encrypted File

Figure A.38, below, illustrates how to read UCMD parameters from an encrypted file. The encrypted file was previously created using the Universal Encrypt utility with the parameters **–userid** and **–pwd**.

```
ucmd -script script.file -host dallas –x encrypted.file
```

Figure A.38 UCMD Manager for Windows - Authentication Parameters from Encrypted File

The command is sent to a remote system named **dallas** for execution. The output of the script execution is redirected back to the UCMD process. Additional command line options are read from the encrypted file **encrypted.file**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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 file from which to read an encrypted command options file. |

# A.3.12 Manager Fault Tolerance

Figure A.39, below, illustrates how to turn manager fault tolerance on. A unique command id is always required for manager fault tolerance.

```
ucmd -script script.file -host dallas -x encrypted.file -managerft yes
-cmdid uniquejobname -restart auto
```

Figure A.39 UCMD Manager for Windows - Manager Fault Tolerance

The command is sent to a remote system named **dallas** for execution. The output of the script is redirected back to the UCMD process. Additional command line options are read from the encrypted file **encrypted.file**. Manager fault tolerance is turned on. A unique command ID is coded to identify the process. Restart is detected automatically. If an executing or pending command ID exists, a reconnect is performed. If not, the process is started as if new.

## Command Line Options

The command line options used in this example are:

| Options | Description |
|---|---|
| -script | Specifies the file 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 file from which to read an encrypted command options file. |
| -managerft | Specifies if the manager fault tolerant feature is used. |
| -cmdid | Specifies the unique command ID associated the unit of work. |
| -restart | Specifies whether or not the manager is requesting restart. |

# A.4 UCMD Manager for UNIX Examples

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

The following list provides a link to each example.

# A.4.1 File Copy Example 1

This example copies a file from a remote Windows system to a local file.

Figure A.40, below, illustrates the command. Although the command is shown on two lines, it should be entered as one line at the command prompt.

```
ucmd -cmd 'ucopy file' -host dallas
     -userid joe -pwd password > localfile
```

Figure A.40 UCMD Manager for UNIX - File Copy Example 1

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 **localfile**.

The command **ucopy** is installed as part of UCMD Server on the remote system.

The file is copied as a text file since the default transfer mode is **text**.

## 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 **dallas**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.4.2 File Copy Example 2

This example copies a local file to a remote Windows system.

Figure A.41, below, illustrates the command. Although the command is shown on two lines, it should be entered as one line at the command prompt.

```
ucmd -cmd 'ucopy > remotefile' -host dallas
     -userid joe -pwd password < localfile
```

Figure A.41 UCMD Manager for UNIX - File Copy Example 2

The **ucopy** command receives its stdin file from ucmd. The standard in of UCMD is redirected from **localfile**.

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 is **text**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ucopy > remotefile** to execute. The **ucopy** program copies its standard in to its standard out. |
| -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. |

## A.4.3 Network Status Script Example

This example produces a report of the network status of a remote Windows system. Instead of executing a command on the remote host, a local script file is executed.

Figure A.42, below, illustrates the script file, **myscript**.

```
echo System Status
echo ------------------------------------------------
date /t
time /t
echo ------------------------------------------------
netstat -se
echo ------------------------------------------------
netstat -a
echo ------------------------------------------------
```

Figure A.42 UCMD Manager for UNIX - Network Status Script File Example

Figure A.43, below, illustrates the command to execute the script file.

```
ucmd -script myscript -host dallas -userid joe -pwd password
```

Figure A.43 UCMD Manager for UNIX - System Status Example

The report is written to the standard out of UCMD.

### Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file name of 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

## A.4.4 Manager Command

Figure A.44, below, illustrates the minimum required parameters for the UCMD Manager, the command to be executed (**-cmd**), and the remote server (**-host**) on which to execute. Security is set to default on the remote server, so the authentication options of **-userid** and **-pwd** also are required.

```
ucmd -cmd 'dir' -host dallas –userid joe –pwd password
```

Figure A.44 UCMD Manager for UNIX - Manager Command

The command **dir** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **dir** command are available to the initiating process. The process will authenticate and run under the authority of userid **joe**. If the remote system is a UNIX system, change the command **dir** to **ls /**.

### Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ls** to execute. |
| -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. |

# A.4.5 Command Coded as a Script

Figure A.45, below, illustrates how to code the remote command as a script.

The benefits are:

- Multiple commands can be executed with one universal command manager.
- Script can be housed separate from the execution command.
- Script can be stored and promoted via change control procedures.
- Provides separation of the remote command from Universal Command options.

```
ucmd -script script.file -host dallas –userid joe –pwd password
```

Figure A.45 UCMD Manager for UNIX - Command Coded as a Script

The contents of **`script.file`** are sent to a remote system named **`dallas`** for execution. The output of the script command is redirected back to the UCMD process. The standard output and standard error of the script execution are available to the initiating process. The process will authenticate and run under the authority of userid **`joe`**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| `-script` | Specifies the file 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`**. |
| `-userid` | Specifies the remote user ID with which to execute the command. |
| `-pwd` | Specifies the password for the user ID. |

# A.4.6 Redirect Standard Out and Standard Error

Figure A.46, below, illustrates how to redirect the standard output and error of the **DIR** command to a file on the initiating system.

```
ucmd -cmd 'dir' -host dallas –userid joe –pwd password > output.file 2>&1
```

Figure A.46 UCMD Manager for UNIX - Redirect Standard Out and Standard Error

The command **dir** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **dir** command are written back to the UCMD process and redirected to standard out file **output.file**. The process will authenticate and run under the authority of userid **joe**.

If the remote system is a UNIX system, change the command **dir** to **ls**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ls** to execute. |
| -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. |

# A.4.7 Redirect Standard Input from Initiating System

The **ucmd** command reads from standard input and writes it to the UCMD Server for the remote command to read as its standard input. The allocation of standard input can be changed with a shell redirection operator. The redirection operators instruct the shell to change the allocation of the standard files. To change the allocation of standard input, use the **<** operator.

```
ucmd –script myscript -host dallas –userid joe –pwd password < input.file
```

Figure A.47 UCMD Manager for UNIX - Redirect Standard Input from Initiating System

The command is sent to a remote system named **dallas** for execution. The output of the script is redirected back to the Universal Command process's standard out and standard error. Standard Input is read from file **input.file** on the initiating system. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.4.8 Redirect Standard Input from /dev/null

If the command ucmd is executed as a background job (using the **&** operator), it will receive the SIGTTIN signal when UCMD tries to read from standard input. Background jobs cannot read their standard input from the terminal since the foreground job (or the shell) has it allocated. The UCMD job is stopped until it is brought to the foreground. To run a UCMD job that does not require terminal input in the background, redirect its standard input from **/dev/null**.

```
ucmd –script &myscript –host dallas  < /dev/null –userid joe –pwd password
```

Figure A.48 UCMD Manager for UNIX - Redirect Standard Input from /dev/null

The command is sent to a remote system named **dallas** for execution. The output of **myscript** is redirected back to the UCMD process. Standard in is read from **/dev/null**. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.4.9 Authentication Parameters from Encrypted File

Figure A.49, below, illustrates how to read UCMD parameters from an encrypted file. The encrypted file was previously created using the Universal Encrypt utility with the parameters **–userid** and **–pwd**.

```
ucmd -script script.file -host dallas –x encrypted.file
```

Figure A.49 UCMD Manager for UNIX - Authentication Parameters from Encrypted File

The command is sent to a remote system named **dallas** for execution. The output of the script execution is redirected back to the UCMD process. Additional command line options are read from the encrypted file **encrypted.file**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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 file from which to read an encrypted command options file. |

# A.4.10 Manager Fault Tolerance

Figure A.50, below, illustrates how to turn on manager fault tolerance. A unique command ID always is required for manager fault tolerance.

```
ucmd -script script.file -host dallas -x encrypted.file -managerft yes
-cmdid uniquejobname -restart auto
```

Figure A.50 UCMD Manager for UNIX - Manager Fault Tolerance

The command is sent to a remote system named **dallas** for execution. The output of the script is redirected back to the UCMD process. Additional command line options are read from the encrypted file **encrypted.file**. Manager fault tolerance is turned on. A unique command ID is coded to identify the process. Restart is detected automatically. If an executing or pending command ID exists, a reconnect is performed. If not, the process is started as if new.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file 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 file from which to read an encrypted command options file. |
| -managerft | Specifies if the manager fault tolerant feature is used. |
| -cmdid | Specifies the unique command ID associated the unit of work. |
| -restart | Specifies whether or not the manager is requesting restart. |

# A.5 UCMD Manager for OS/400 Examples

This section contains examples demonstrating the use of:

- UCMD Manager for OS/400
- UCMD Managers on UNIX and Windows to interact with OS/400

The following list provides a link to each example.



These examples reference the OS/400 commands by their untagged names. If you are using commands with tagged names to run UCMD, substitute the tagged names for the untagged names. (See Section 6.2.1 Universal Products for OS/400 Commands for further information.)

# A.5.1 File Copy Example 1

This example copies a file from a remote Windows system to a local file.

Figure A.51, below, illustrates the command.

```
STRUCM CMD('ucopy file') HOST(dallas)
     USERID(joe) PWD(password) SOTFILE(localfile)
```

Figure A.51 UCMD Manager for OS/400 - File Copy Example 1

The standard out of the **ucopy** command on the remote host is redirected back to the local host and written to the file specified by **SOTFILE**.

The command **ucopy** is installed as part of UCMD Server on the remote system.

The file is copied as a text file, since the default transfer mode is **text**.

## 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 **dallas**. |
| USERID | Specifies the remote user ID with which to execute the command. |
| PWD | Specifies the password for the user ID. |
| SOTFILE | Specifies the local file that receives the stdout of the remote command. |

# A.5.2 File Copy Example 2

This example copies a local file to a remote Windows system.

Figure A.52, below, illustrates the command.

```
STRUCM CMD('ucopy > remotefile') HOST(dallas)
     USERID(joe) PWD(password) SINFILE(localfile)
```

Figure A.52 UCMD Manager for OS/400 - File Copy Example 2

The **ucopy** command receives its standard in file from STRUCM. The standard in of STRUCM is read from **localfile**, as specified by **SINFILE**.

The command **ucopy** is installed as part of UCMD Server on the remote system.

The file is copied as a text file since the default transfer mode is **text**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| CMD | Specifies the remote command **ucopy > remotefile** to execute. The **ucopy** program copies its stdin 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. |

# A.5.3 File Copy Example 3

This example shows a file copy initiated by a Windows system which copies a file from the OS/400 to the Windows system.

Figure A.53, below, illustrates the command.

```
ucmd -i sysName -u userId -w password -c "strucp frmfile(mylib/myfile)" >
D:\tmp\File400.txt
```

Figure A.53 UCMD Manager for OS/400 - File Copy Example 3

The **ucopy** command receives its stdin via the **TOFILE** parameter. (The **TOFILE** parameter is not shown here, since it defaults to standard out.) Of course, the OS/400-side standard out maps to the Windows system standard in, since the data is transferred from the OS/400 (output) system to the Windows system (input). The Windows system then transfers the data input from the OS/400 system to a file, **D:\tmp\File400.txt**, on the Windows system.

The command **strucp** is installed as part of UCMD Server on the OS/400 system.

The file is copied as a text file, since the default transfer mode is **text**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -c (-cmd) | Specifies the remote command to execute on the OS/400. |
| -i (-host) | Directs the command to a computer with a host name of **sysName**. |
| -u (-userid) | Specifies the OS/400 user ID with which to execute the command. |
| -w (-pwd) | Specifies the password for the user ID. |

# A.5.4 File Copy Example 4

This example shows a file copy initiated by a Windows system which copies a file from the OS/400 to the Windows system.

Figure A.54, below, illustrates the command.

```
ucmd -i sysName -u userId -w password -c "strucp tofile(mylib/readme)" <
D:\tmp\README.txt
```

Figure A.54 UCMD Manager for OS/400 - File Copy Example 4

The **strucp** command receives its standard in via the **FRMFILE** parameter (**mylib/myfile**). (The **FRMFILE** parameter is not shown, since it defaults to standard in as its input source.) In this case, the standard out, output data stream from the Windows system is mapped to the OS/400-side standard in data stream. The OS/400 system then transfers the data input from the Windows system to a file, **mylib/readme**, on the OS/400 system.

The command **strucp** is installed as part of UCMD Server on the OS/400 system.

The file is copied as a text file since the default transfer mode is **text**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -c (-cmd) | Specifies the remote command to execute on the OS/400. |
| -i (-host) | Directs the command to a computer with a host name of **sysName**. |
| -u (-userid) | Specifies the OS/400 user ID with which to execute the command. |
| -w (-pwd) | Specifies the password for the user ID. |

# A.5.5 Display Library with Manager Fault Tolerance Active Using USBMJOB Example

Figure A.55, below, illustrates the use of an OS/400 command on a remote system with spooling enabled.

It assumes that manager fault tolerance is active on the client platform via the UCMD configuration file. The example should execute from either a UNIX shell or a Windows system environment. The command is submitted via **USBMJOB** to allow the output data and the job log of the executed command to be brought back to the system initiating the command.

```
Windows System
ucmd -c "USBMJOB CMD(dsplib)" -u userId -w password -i sysName
    -cmdid NTSysTest <NUL 1>out400.txt 2>err400.txt

UNIX System
ucmd -c "USBMJOB CMD(dsplib qsysopr)" -u userId -w userPW
    -i sysName -cmdid UNIX00 1>out400.txt 2>err400.txt
```

Figure A.55 UCMD Manager for OS/400 - Display Library

For this example, **USBMJOB** requires no input; however, the user must supply **<NUL** to satisfy Windows operating system requirements. Without **<NUL**, the request will hang. Data and job logs are sent out from the OS/400 via standard out and standard error, respectively. The OS/400 also sends any error messages via standard error. The system takes data sent back to UCMD and stores it in **out400.txt**; it takes any error messages and the job logs and stores them in **err400.txt**.

The command **USBMJOB** is installed as part of UCMD Server on the OS/400 system.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -c (-cmd) | Specifies the remote command to execute on the OS/400. |
| -i (-host) | Directs the command to a computer with a host name of **dallas**. |
| -u (-userid) | Specifies the OS/400 user ID with which to execute the command. |
| -w (-pwd) | Specifies the password for the user ID. |
| -cmdid (-C) | UCMD Server (running under UBroker) puts the Command ID in its database to keep track of requests regarding a specific unit of work. |

Note: Include the command option **-managerft yes**, requesting manager fault tolerance, if it is not enabled via the UCMD configuration file.

# A.5.6 Network Status Script Example

This example produces a report of the network status of a remote Windows system. Instead of executing a command on the remote host, a local script file is executed.

Figure A.56, below, illustrates the script file, **MYSCRIPT**.

```
echo System Status
echo ------------------------------------------------
date /t
time /t
echo ------------------------------------------------
netstat -se
echo ------------------------------------------------
netstat -a
echo ------------------------------------------------
```

Figure A.56 UCMD Manager for OS/400 - Network Status Script File

Figure A.57, below, illustrates the command to execute the script file.

```
STRUCM SCRIPT(myscript) HOST(dallas) USERID(joe)
PWD(password)
```

Figure A.57 UCMD Manager for OS/400 - Network Status Script File Command

The report is written to the stdout of **STRUCM**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| SCRIPT | Specifies the file name of 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**. |
| USERID | Specifies the remote user ID with which to execute the command. |
| PWD | Specifies the password for the user ID. |

# A.5.7 Manager Command

The OS/400 Manager provides a command line interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered them directly on the remote command line. Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

UCMD Manager for OS/400 executes as a CL command.

Figure A.58, below, illustrates the minimum required parameters for the UCMD Manager, the command to be executed (CMD), and the remote server (HOST) on which to execute.

```
STRUCM CMD('ls -al') HOST(dallas)
```

Figure A.58 UCMD Manager for OS/400 - Manager Command

The command **`ls -al`** is sent to a remote system named **`dallas`** for execution. The standard output and standard error of the **`ls`** command are available to the initiating process and written to standard out (**`QPRINT`**).

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| CMD | Specifies the remote command **`ls`** to execute. |
| HOST | Directs the command to a computer with a host name of **`dallas`**. |

# A.5.8 Copy File from Remote Windows to Local OS/400

Figure A.59, below, illustrates the copying of a file from a remote Windows system to a local file.

```
STRUCM CMD('ucopy file') HOST(dallas) USERID(joe) PWD(password)
SOTFILE(localfile)
```

Figure A.59 UCMD Manager for OS/400 - Copy Files from Remote Windows to Local OS/400

The command **ucopy file** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **ucopy** command are available to the initiating process as file **localfile**. The command **ucopy** is installed as part of UCMD Server on the remote system.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| CMD | Specifies the remote command **ucopy file** to execute. |
| 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. |
| SOTFILE | Specifies the local file that receives the stdout of the remote command. |

# A.5.9 Command Coded as a Script

Figure A.60, below, illustrates the execution of a network status script on a remote Windows server.

```
STRUCM SCRIPT(myscript) HOST(dallas) USERID(joe)
PWD(password)
```

Figure A.60 UCMD Manager for OS/400 - Command Coded as a Script

The command **myscript** is sent to a remote system named **dallas** for execution. The standard output and standard error of **myscript** command are available to the initiating process as file **QPRINT**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| CMD | Specifies the remote command **ucopy file** to execute. |
| 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. |

# A.6 UCMD Manager for HP NonStop Examples

This section contains examples demonstrating the use of UCMD Manager for HP NonStop.

The following list provides a link to each example.

# A.6.1 File Copy Example 1

Figure A.61, below, illustrates the command that copies a file from a remote NT system to a local file.

Although the command is shown on two lines, it should be entered on one line at the command prompt.

```
run ucmd –cmd 'ucopy file' -host dallas –server “ –script_type OSS”
     -userid joe –pwd password –stdout –localfile localfile
```

Figure A.61 UCMD Manager for HP NonStop - File Copy Example 1

The stdout of the **ucopy** command on the remote host is redirected back to the local host and written to the standard out of UCMD, which is then redirected to the local file **localfile**.

The command **ucopy** is installed as part of UCMD Server on the remote system.

The file is copied as a text file, since the default transfer mode is **text**.

## 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 **dallas**. |
| -server | Specifies the command lines options for the UCMD Server process. The value **–script_type OSS** is specified to notify the UCMD Server that it is to execute an OSS process, since universal copy is a native OSS program. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -stdout | Specifies the start of the stdout options. |
| -localfile | Specifies the filename to which to redirect output. |

# A.6.2 File Copy Example 2

This example copies a local file to a remote Windows system.

Figure A.62, below, illustrates the command. Although the command is shown on multiple lines, it should be entered as one line at the command prompt.

```
run ucmd -cmd 'ucopy > remotefile' -host dallas
     -server " -script_type OSS" -userid joe -pwd password
     -stdin -localfile localfile
```

Figure A.62 UCMD Manager for HP NonStop - File Copy Example 2

The **ucopy** command receives its standard in file from UCMD. The standard in of UCMD is redirected from **localfile**.

The command **ucopy** is installed as part of UCMD Server on the remote system.

The file is copied as a text file since the default transfer mode is **text**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ucopy > remotefile** to execute. The **ucopy** program copies it stdin to its stdout. |
| -host | Directs the command to a computer with a host name of **dallas**. |
| -server | Specifies the command lines options for the UCMD Server process. The value **-script_type OSS** is specified to notify the UCMD Server that it is to execute an OSS process, since universal copy is a native OSS program. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |
| -stdout | Specifies the start of the stdout options. |
| -localfile | Specifies the filename to which to redirect output. |

# A.6.3 Network Status Report Example

This example produces a report of the network status of a remote Windows system. Instead of executing a command on the remote host, a local script file is executed.

Figure A.63, below, illustrates the script file, **myscript**.

```
echo System Status
echo ------------------------------------------------
date /t
time /t
echo ------------------------------------------------
netstat -se
echo ------------------------------------------------
netstat -a
echo ------------------------------------------------
```

Figure A.63 UCMD Manager for HP NonStop - Network Status Script File

Figure A.64, below, illustrates the command to execute the script file.

```
run ucmd -script myscript -host dallas -userid joe -pwd password
```

Figure A.64 UCMD Manager for HP NonStop - System Status Example

The report is written to the standard out of UCMD.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -script | Specifies the file name of 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**. |
| -userid | Specifies the remote user ID with which to execute the command. |
| -pwd | Specifies the password for the user ID. |

# A.6.4 Manager Command

The UCMD Manager for HP NonStop provides a command line interface to remote computers running the UCMD Server component. The UCMD Manager executes remote commands as they would be if you entered the command directly on the remote command line. Remote standard input and output files are redirected to the UCMD Manager's standard input and output files.

Figure A.65, below, illustrates the minimum required parameters for the UCMD Manager, the command to be executed (**-cmd**), and the remote server (**-host**) on which to execute. Authentication is turned off on the remote system.

```
run $SYSTEM.UNVBIN.ucmd –cmd 'ls' -host dallas –userid joe –pwd password
```

Figure A.65 UCMD Manager for HP NonStop - Manager Command

The command **ls** is sent to a remote system named **dallas** for execution. The standard output and standard error of the **ls** command are available to the standard out of the initiating process. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command to execute. |
| -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. |

# A.6.5 Copy Remote Windows File to Local HP NonStop

Figure A.66, below, illustrates how to copy a file from a remote Windows system to a local file.

Although the command is shown on two lines, it should be entered on one line at the command prompt.

```
run $SYSTEM.UNVBIN.ucmd –cmd 'ucopy file' -host dallas -userid joe
-pwd password –stdout –localfile localfile
```

Figure A.66 UCMD Manager for HP NonStop - Copy Remote Windows File to Local HP NonStop

The standard out of the **ucopy** command on the remote host is redirected back to the local host and written to the standard out of UCMD, which then is redirected to the local file **localfile**. The command **ucopy** is installed as part of UCMD Server on the remote system. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ucopy file** to execute. |
| -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. |
| -stdout | Specifies the start of the stdout options. |
| -localfile | Specifies the filename to which to redirect output. |

# A.6.6 Copy Local File to Remote Windows

Figure A.67, below, illustrates how to copy 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.

```
run $SYSTEM.UNVBIN.ucmd –cmd 'ucopy > remotefile' -host dallas -userid joe
–pwd password -stdin –localfile localfile
```

Figure A.67 UCMD Manager for HP NonStop - Copy Local File to Remote Windows

The **ucopy** command receives its standard in file from the UCMD **localfile** parameter. The file is written to **remotefile** on the remote system. The command **ucopy** is installed as part of UCMD Server on the remote system. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ucopy file** to execute. |
| -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. |
| -stdin | Specifies the start of the stdin options. |
| -localfile | Specifies the filename from which to redirect input. |

# A.6.7 Command Coded as a Script

This example executes a network status script on a remote Windows server.

```
run $SYSTEM.UNVBIN.ucmd –script myscript -host dallas -userid joe
–pwd password
```

Figure A.68 UCMD Manager for HP NonStop - Command Coded as a Script

The command **myscript** is sent to a remote system named **dallas** for execution. The standard output and standard error of **myscript** command are available to the standard out of the initiating process. The process will authenticate and run under the authority of userid **joe**.

## Command Line Options

The command line options used in this example are:

| Option | Description |
|---|---|
| -cmd | Specifies the remote command **ucopy file** to execute. |
| -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. |

# Appendix B
# Customer Support

Stonebranch, Inc. provides customer support, via telephone and e-mail, for Universal Command (UCMD) 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**

**950 North Point Parkway, Suite 200**
**Alpharetta, Georgia 30005**
**U.S.A.**