



Fermilab

GU0014A

Complete Guide and Reference Manual for UPS, UPD and UPP v4

Part I: Overview and End User's Guide and Part VI: UPS and UPD Command Reference

Release 2.0

June 30, 2000

Computing Division
Fermi National Accelerator Laboratory

Compiled by Anne Heavey

ABSTRACT

This manual documents the standard methodology for UNIX product support at Fermilab, which is implemented via the utilities **UPS** (UNIX Product Support), **UPD** (UNIX Product Distribution), and **UPP** (UNIX Product Poll). These utilities were significantly redesigned for version v4, which was initially released in 1998, and have continued to be revised since then. The latest release as of this writing is v4_5_2. This document supersedes GU0014 "UPS and UPD v4 Reference Manual", released June 5, 1998.

This part of the document (GU0014A) includes a guide for end users and a command reference.

Revision Record

May 1997	Original Release 1.0 (for UPS v3 and UPD v2)
August 1997	Revisions 1.1 and 1.1a (for UPS v3 and UPD v2)
June 1998	Release 1.0 for UPS and UPD v4
December 1999	Draft release 2.0 for UPS/UPD/UPP v4. Part VI Command Reference only
June 2000	Release 2.0 for UPS, UPD and UPP v4 (current as of v4_5_2)

This document and associated documents and programs, and the material and data contained therein, were developed under the sponsorship of an agency of the United States government, under D.O.E. Contract Number EY-76-C-02-3000 or revision thereof. Neither the United States Government nor the Universities Research Association, Inc. nor Fermilab, nor any of their employees, nor their respective contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights. Mention of any specific commercial product, process, or service by trade name, trademark, manufacturer, supplier, or otherwise, shall not, nor is it intended to, imply fitness for any particular use, or constitute or imply endorsement, recommendation, approval or disapproval by the United States Government or URA or Fermilab. A royalty-free, non-exclusive right to use and disseminate same for any purpose whatsoever is expressly reserved to the U.S. and the U.R.A. Any further distribution of this software or documentation, parts thereof, or other software or documentation based substantially on this software or parts thereof will acknowledge its source as Fermilab, and include verbatim the entire contents of this Disclaimer, including this sentence.

Acknowledgments

The redesign and redevelopment of **UPS** and its companion products in preparation for Fermilab's Run II involved a substantial commitment of resources from the Computing Division in 1997-98. Special thanks to Don Petravick (HPPC), Ruth Pordes (OLS), and Dane Skow (OSS) for providing talented and motivated members of their groups to accomplish this task. Since the initial release of **UPS/UPD v4** in 1998, development has been continuing, and we are at version v4_5_2 as of this writing.

The redevelopment effort was led by Eileen Berman. With her, the principal designers and developers of **UPS/UPD v4** included David Fagan, Marc Mengel, Lars Rasmussen and Margaret Votava. Other contributors to the new design included Lauri Loebel Carpenter, Rob Harris, Alan Jonckheere, Art Kreymer, Liz Sexton-Kennedy. Other contributors to the coding effort included Chuck Debaun, Paul Russo and Don Walsh.

Contributors in the areas of code review, testing, documentation review and deployment included Lauri Loebel Carpenter, Chuck Debaun, Lisa Giacchetti, Alan Jonckheere, Art Kreymer, Liz Sexton-Kennedy, Mike Stolz, Don Walsh and Gordon Watts, in addition to the development team. Special thanks go to Marc Mengel and Margaret Votava for contributing all the updated **UPD** and **UPP** information included in the first release of this manual for **UPS/UPD v4**.

Wayne Baisley and Marc Mengel are currently responsible for on-going support and development of **UPS/UPD**, and thanks go to them for providing quite a bit of updated information for this release of the manual. Thanks are also due to Wayne and Marc as well as to Joy Hathaway, Lauri Loebel Carpenter and Cindy Wike for reviewing portions of the documentation and providing feedback.

Table of Contents for Parts I and VI

About this Manual	INT-1
Document Structure, Purpose and Intended Audiences.....	INT-1
Availability	INT-3
Updates.....	INT-3
Conventions	INT-3
Your Comments are Welcome!	INT-5

Part I: Overview and End User's Guide

Chapter 1: Overview of UPS, UPD and UPP v4	1-1
1.1 Introduction to UPS, UPD and UPP	1-1
1.2 Motivation for the UPS Methodology	1-2
1.3 UPS Products	1-3
1.3.1 Versions	1-3
1.3.2 Flavors	1-3
1.3.3 Qualifiers	1-4
1.3.4 Product Instances	1-4
1.3.5 Chains	1-4
1.3.6 Product Dependencies	1-5
1.3.7 Product Overlays	1-6
1.4 UPS Database Overview	1-6
1.4.1 UPS Database Files	1-6
1.4.2 UPS Database Structure	1-7
1.5 Using UPS Without a Database	1-7
1.6 UPS and UPD Commands	1-8
1.6.1 Syntax	1-8
1.6.2 Defaults	1-8
1.7 The UPS Environment	1-9
1.7.1 Initializing the UPS Environment	1-9
1.7.2 Changes UPS Makes to your Environment	1-10

Chapter 2: UPS Operations for the End User	2-1
2.1 Determining your Machine's Flavor	2-1
2.2 Listing Product Information in a Database	2-2
2.2.1 Formatted Output Style	2-3
2.2.2 Condensed Output Style	2-3
2.2.3 Examples	2-4
2.3 Finding a Product's Dependencies	2-7
2.4 Setting up a Product	2-8
2.4.1 The setup Command for the Typical Case	2-9
2.4.2 When You Need to Specify Other Options	2-9
2.5 Running Unsetup on a Product	2-10

Part VI: UPS and UPD Command Reference

Chapter 22: UPS Command Reference	22-1
22.1 setup	22-3
22.1.1 Command Syntax	22-3
22.1.2 Commonly Used Options	22-3
22.1.3 All Valid Options	22-3
22.1.4 More Detailed Description	22-5
22.1.5 setup Examples	22-6
22.2 unsetup	22-9
22.2.1 Command Syntax	22-9
22.2.2 All Valid Options	22-9
22.2.3 More Detailed Description	22-11
22.2.4 unsetup Examples	22-12
22.3 ups configure	22-13
22.3.1 Command Syntax	22-13
22.3.2 Commonly Used Options	22-13
22.3.3 All Valid Options	22-13
22.3.4 More Detailed Description	22-15
22.3.5 ups configure Examples	22-15
22.4 ups copy	22-17
22.4.1 Command Syntax	22-17
22.4.2 Commonly Used Options	22-17
22.4.3 All Valid Options	22-17
22.4.4 Options Valid with -G	22-19
22.4.5 More Detailed Description	22-19
22.4.6 ups copy Examples	22-20
22.5 ups declare	22-21
22.5.1 Command Syntax	22-21
22.5.2 Commonly Used Options	22-21
22.5.3 All Valid Options	22-22

22.5.4	More Detailed Description	22-24
22.5.5	ups declare Examples	22-26
22.6	ups depend	22-27
22.6.1	Command Syntax	22-27
22.6.2	Commonly Used Options	22-27
22.6.3	All Valid Options	22-27
22.6.4	ups depend Examples	22-29
22.7	ups exist	22-31
22.7.1	Command Syntax	22-31
22.7.2	Commonly Used Options	22-31
22.7.3	All Valid Options	22-31
22.7.4	More Detailed Description	22-33
22.7.5	ups exist Examples	22-33
22.8	ups flavor	22-35
22.8.1	Command Syntax	22-35
22.8.2	Commonly Used Options	22-35
22.8.3	All Valid Options	22-35
22.8.4	More Detailed Description	22-36
22.8.5	ups flavor Examples	22-37
22.9	ups get	22-39
22.9.1	Command Syntax	22-39
22.9.2	All valid options	22-39
22.9.3	ups get Example	22-40
22.10	ups help	22-41
22.10.1	ups help Example	22-41
22.11	ups list	22-43
22.11.1	Command Syntax	22-43
22.11.2	Commonly Used Options	22-43
22.11.3	All Valid Options	22-43
22.11.4	More Detailed Description	22-45
22.11.5	ups list Examples	22-49
22.12	ups modify	22-55
22.12.1	Command Syntax	22-55
22.12.2	Commonly Used Options	22-55
22.12.3	All Valid Options	22-55
22.12.4	More Detailed Description	22-56
22.12.5	ups modify Example	22-57
22.13	ups start	22-59
22.13.1	Command Syntax	22-59
22.13.2	Commonly Used Options	22-59
22.13.3	All Valid Options	22-59
22.13.4	More Detailed Description	22-61
22.13.5	ups start Examples	22-61
22.14	ups stop	22-63
22.14.1	Command Syntax	22-63
22.14.2	Commonly Used Options	22-63

22.14.3	All Valid Options	22-63
22.14.4	More Detailed Description	22-65
22.14.5	ups stop Examples	22-65
22.15	ups tailor	22-67
22.15.1	Command Syntax	22-67
22.15.2	Commonly Used Options	22-67
22.15.3	All Valid Options	22-67
22.15.4	More Detailed Description	22-69
22.15.5	ups tailor Example	22-69
22.16	ups touch	22-71
22.16.1	Command Syntax	22-71
22.16.2	Commonly Used Options	22-71
22.16.3	All Valid Options	22-71
22.16.4	ups touch Example	22-72
22.17	ups unconfigure	22-73
22.17.1	Command Syntax	22-73
22.17.2	Commonly Used Options	22-73
22.17.3	All Valid Options	22-73
22.17.4	More Detailed Description	22-75
22.17.5	ups unconfigure Example	22-75
22.18	ups undeclare	22-77
22.18.1	Command Syntax	22-77
22.18.2	Commonly Used Options	22-77
22.18.3	All Valid Options	22-78
22.18.4	More Detailed Description	22-79
22.18.5	ups undeclare Examples	22-80
22.19	ups verify	22-81
22.19.1	Command Syntax	22-81
22.19.2	Commonly Used Options	22-81
22.19.3	All Valid Options	22-81
22.19.4	ups verify Example	22-83
Chapter 23:	UPD/UPP Command Reference	23-1
23.1	upd addproduct	23-3
23.1.1	Command Syntax	23-3
23.1.2	Commonly Used Options	23-4
23.1.3	All Valid Options	23-4
23.1.4	More Detailed Description	23-7
23.1.5	Adding Products to fnkits.fnal.gov	23-8
23.1.6	upd addproduct Examples	23-9
23.2	upd cloneproduct	23-11
23.2.1	Command Syntax	23-11
23.2.2	All Valid Options	23-11
23.2.3	Options Valid with -G	23-12
23.2.4	upd cloneproduct Example	23-12

23.3	upd delproduct	23-13
23.3.1	Command Syntax	23-13
23.3.2	Commonly Used Options	23-13
23.3.3	All Valid Options	23-13
23.3.4	upd delproduct Example	23-14
23.4	upd depend	23-15
23.4.1	Command Syntax	23-15
23.4.2	Options	23-15
23.4.3	upd depend Examples	23-15
23.5	upd exist	23-17
23.5.1	Command Syntax	23-17
23.5.2	Options	23-17
23.5.3	upd exist Examples	23-17
23.6	upd fetch	23-19
23.6.1	Command Syntax	23-19
23.6.2	Commonly Used Options	23-19
23.6.3	All Valid Options	23-19
23.6.4	upd fetch Examples	23-21
23.7	upd get	23-23
23.7.1	Command Syntax	23-23
23.7.2	Options	23-23
23.8	upd install	23-25
23.8.1	Command Syntax	23-25
23.8.2	Commonly Used Options	23-25
23.8.3	All Valid Options	23-25
23.8.4	Options Valid with -G	23-28
23.8.5	More Detailed Description	23-28
23.8.6	upd install Examples	23-29
23.9	upd list	23-31
23.9.1	Command Syntax	23-31
23.9.2	Options	23-31
23.9.3	upd list Examples	23-31
23.10	upd modproduct	23-33
23.10.1	Command Syntax	23-33
23.10.2	Commonly Used Options	23-33
23.10.3	All Valid Options	23-34
23.10.4	More Detailed Description	23-35
23.10.5	upd modproduct Examples	23-36
23.11	upd reproduct	23-39
23.11.1	Command Syntax	23-39
23.11.2	Options	23-40
23.11.3	upd reproduct Examples	23-40
23.12	upd update	23-41
23.12.1	Command Syntax	23-41
23.12.2	Commonly Used Options	23-41

23.12.3	All Valid Options	23-41
23.12.4	upd update Examples	23-43
23.13	upd verify	23-45
23.13.1	Command Syntax	23-45
23.13.2	Options	23-45
23.14	upp	23-47
23.14.1	Command Syntax	23-47
23.14.2	All Valid Options	23-47
23.14.3	upp Examples	23-47
Chapter 24:	Generic Command Option Descriptions	24-1
24.1	Alphabetical Option Listing	24-1
24.2	More Information on Selected Options	24-7
24.2.1	-e	24-7
24.2.2	-H	24-7
24.2.3	-K	24-7
24.2.4	-q	24-8
24.2.5	-V	24-9
Chapter 25:	UPS/UPD Command Usage	25-1
25.1	Syntax	25-1
25.1.1	Order of Command Line Elements	25-1
25.1.2	Specifying Version/Chain	25-1
25.1.3	Grouping Option Flags	25-2
25.1.4	Specifying Arguments to Options	25-2
25.1.5	Embedded Spaces in Option Arguments	25-2
25.1.6	Invalid Option Arguments	25-3
25.1.7	Specifying Multiple Products in a Single Command	25-3
25.1.8	Multiple Occurrences of Same Option Flag	25-3
25.1.9	Use of Wildcards	25-4
25.2	Options	25-4
Chapter 26:	Product Instance Matching in UPS/UPD Commands	26-1
26.1	Database Selection Algorithm	26-1
26.1.1	UPS	26-1
26.1.2	UPD	26-2
26.2	Instance Matching within Selected Database	26-3
26.2.1	Where Does Instance Matching Take Place?	26-3
26.2.2	Flavor Selection	26-3
26.2.3	Qualifiers: Use in Instance Matching	26-4
26.2.4	Flavor and Qualifier Matching Algorithm	26-4
Glossary		GLO-1
Index		IDX-1

Table of Contents for Complete Guide

About this Manual	INT-1
--------------------------------	--------------

(This introductory chapter is listed in the front section of the table of contents.)

Part I: Overview and End User's Guide

(Part I is listed is listed in the front section of the table of contents.)

Chapter 1: Overview of UPS, UPD and UPP v4	1-1
---	------------

Chapter 2: UPS Operations for the End User	2-1
---	------------

Part II: Product Installer's Guide

Chapter 3: General Product Installation Information	3-1
--	------------

3.1 Installation Methods for UPS Products	3-1
---	-----

3.1.1 UPD	3-1
-----------------	-----

3.1.2 UPP	3-2
-----------------	-----

3.1.3 FTP	3-2
-----------------	-----

3.2 User Node Registration for KITS	3-2
---	-----

3.3 What You Need to Know about Your System's UPD Configuration ..	3-3
--	-----

3.3.1 Location of UPD Configuration File	3-3
--	-----

3.3.2 Where Products Get Declared	3-4
---	-----

3.3.3 Where Products Get Installed	3-4
--	-----

3.4 Declaring an Instance Manually	3-5
--	-----

3.4.1 The ups declare Command	3-5
-------------------------------------	-----

3.4.2 Examples	3-6
----------------------	-----

3.5 Installation FAQ	3-7
----------------------------	-----

3.5.1 What File Permissions Get Set?	3-7
--	-----

3.5.2 You're Ready to Install: Should you Declare Qualifiers?	3-8
---	-----

3.5.3 What if an Install Gets Interrupted?	3-8
--	-----

3.5.4 What if a Product was Installed under a Different Name?	3-8
---	-----

3.6 Post-Installation Procedures	3-9
3.6.1 Configuring a Product	3-9
3.6.2 Tailoring a Product	3-9
3.7 Networking Restrictions at your Site	3-9
3.7.1 Proxying Webserver	3-9
3.7.2 Firewall for Incoming TCP Connections	3-10
Chapter 4: Finding Information about Products on a Distribution Node .	4-1
4.1 Listing Products on a Distribution Node	4-1
4.1.1 Using UPD	4-1
4.1.2 Using UPP	4-3
4.2 Listing Product Dependencies on a Distribution Node	4-5
4.3 Information about Products in KITS	4-6
4.3.1 Access Restrictions and Product Categories	4-6
4.3.2 Product Pathnames for FTP Access	4-7
4.4 Special Instructions for Proprietary Products	4-8
Chapter 5: Installing Products Using UPD	5-1
5.1 The upd install Command	5-1
5.1.1 Command Syntax	5-1
5.1.2 Passing Options to the Local ups declare Command	5-2
5.2 How UPD Selects the Database	5-2
5.2.1 Database Selection Algorithm	5-2
5.2.2 Database Selection for Dependencies	5-3
5.2.3 Selecting a Database for Development or Testing	5-3
5.3 Checklist for Installing a Product using UPD	5-3
5.4 Examples	5-4
5.4.1 Install a Product Using Default Database	5-4
5.4.2 Install a Product, Specifying Database	5-5
5.4.3 Install a Product and All Dependencies	5-5
5.4.4 Install a Product and No Dependencies	5-7
5.4.5 Install a Product and Required Dependencies Only	5-7
Chapter 6: Installing Products Using UPP	6-1
6.1 Overview of Using UPP to Install Products	6-1
6.2 Creating a UPP Subscription File	6-1
6.2.1 Create the Header	6-2
6.2.2 Identify the Product	6-2
6.2.3 Trigger the Product Installation	6-2
6.2.4 Provide Instructions to UPP	6-3
6.3 Sample Subscription File for Installing a Product	6-3
6.4 The UPP Command	6-4
6.5 Automating UPP via cron	6-4
Chapter 7: Installing Products using FTP	7-1
7.1 UPS Product Components to Download	7-1
7.2 Installing Products from fnkits.fnal.gov	7-2
7.2.1 Download the Files from fnkits	7-2

7.2.2	Unwind the Files into your Products Area	7-3
7.2.3	Declare the Product to your Database	7-4
7.3	Installing Products from Other Product Distribution Nodes	7-4
7.3.1	Locate the Product Files on the Server	7-4
7.3.2	Download the Files from the Server	7-5
7.3.3	Unwind the Files into your Products Area	7-5
7.3.4	Declare the Product to your Database	7-5
Chapter 8:	Product Installation: Special Cases	8-1
8.1	Installing Products that Require Special Privileges	8-1
8.2	Installing Locally Using UPD from AFS-Space	8-2
8.3	Installing Products into AFS Space	8-3
8.3.1	Overview	8-3
8.3.2	Request a Product Volume	8-4
8.3.3	Install your Product	8-4
8.3.4	Post-Installation Steps	8-5
Chapter 9:	Troubleshooting UPS Product Installations	9-1

Part III: System Administrator's Guide

Chapter 10:	Maintaining a UPS Database	10-1
10.1	Declare an Instance	10-1
10.1.1	The ups declare Command	10-2
10.1.2	Examples	10-2
10.2	Declare a Chain	10-4
10.2.1	The ups declare Command with Chain Specification	10-4
10.2.2	Examples	10-5
10.3	Remove a Chain	10-6
10.4	Change a Chain	10-7
10.5	Undeclare and Remove an Instance	10-7
10.5.1	Using ups undeclare to Remove a Product	10-8
10.5.2	Undoing Configuration Steps	10-9
10.5.3	Using UPP to Remove a Product	10-10
10.6	Verify Integrity of an Instance	10-10
10.7	Modify Information in a Database File	10-11
10.8	Determine If a Product Needs to be Updated	10-13
10.8.1	Using UPP	10-13
10.8.2	Using UPD	10-13
10.9	Update a Table File or ups Directory	10-14
10.10	Retrieve an Individual File	10-15
10.11	Check Product Accessibility	10-16
10.12	Troubleshooting	10-17

Chapter 11: UPS and UPD Pre-install Issues and General Administration	11-1
11.1 Choosing Installer Accounts	11-1
11.1.1 Single Installer Account	11-1
11.1.2 Multiple Installer Accounts	11-1
11.1.3 Separate Installer Accounts for Different Product Categories	11-2
11.2 Setting gids for Multiple Installer Accounts	11-2
11.3 File Ownership, Permissions and Access Restrictions	11-3
11.3.1 Product Files	11-3
11.3.2 Database Files	11-3
11.4 Product File Location and Organization	11-4
11.4.1 Considerations	11-4
11.4.2 Single Flavor or Single Node Systems	11-4
11.4.3 Multi-Flavor and/or Multi-Node Systems	11-5
11.5 Database File Location and Organization	11-6
11.5.1 Choosing Single or Multiple UPS Databases	11-6
11.5.2 UPS Database File Pointers	11-6
11.6 Installing UPS for Use Without a Database	11-7
11.7 CYGWIN (Windows NT) Issues	11-7
11.7.1 Using Correct Perl Version	11-7
11.7.2 Mounting the CYGWIN bin Directory	11-8
11.7.3 Setting Environment Variables	11-8
11.8 General Administration Issues	11-8
11.8.1 Upgrading an Older System	11-8
11.8.2 Adding a New Database and/or Products Area	11-9
11.8.3 Collecting Statistics on Product Usage	11-10
Chapter 12: Providing Access to AFS Products	12-1
12.1 Overview	12-1
12.2 Configuring a Local Database to Work With AFS	12-2
12.2.1 Steps to Create and Configure the Database	12-2
12.2.2 Post-Configuration: Reinitialize FUE Environment	12-4
12.2.3 A Note about Product Installation for this Configuration	12-4
12.3 Installing a Local Copy of CoreFUE	12-4
12.4 Additional Steps for Unfamiliar Naming Conventions	12-5
12.5 Updating /usr/local/bin to Access AFS Products	12-6
Chapter 13: Bootstrapping CoreFUE	13-1
13.1 Downloading the Bootstrap and Configuration Files	13-1
13.1.1 Predefined Configurations for UNIX	13-1
13.1.2 User-defined Configuration for UNIX	13-2
13.1.3 Predefined Configurations for NT	13-2
13.2 Customizing a Bootstrap Configuration	13-3
13.2.1 Bootstrap Configuration File Statement Definitions	13-3
13.2.2 Sample Customization	13-4

13.3 Running the Bootstrap Procedure	13-5
13.3.1 UNIX	13-5
13.3.2 NT	13-5
Chapter 14: Automatic UPS Product Startup and Shutdown	14-1
14.1 Configuring Your Machine to Allow Automatic Startup/Shutdown .	14-1
14.2 Installing a UPS Product to Start and/or Stop Automatically	14-2
14.2.1 Determine if Auto Start/Stop Feature is Enabled	14-2
14.2.2 Determine if Product is Appropriate for Autostart	14-3
14.2.3 Edit Control File(s)	14-3
14.2.4 Summary	14-4
14.3 Disabling UPS Automatic Start/Stop of Processes	14-4
14.4 A Summary of the UPS Automatic Start-up Process	14-5

Part IV: Product Developer's Guide

Chapter 15: UPS Product Development: General Considerations	15-1
15.1 Product Development Considerations and Recommendations	15-1
15.1.1 All Products (Locally Developed and Third Party)	15-1
15.1.2 Products that You Develop	15-2
15.1.3 Third-Party Products Requiring a Hard-Coded Path	15-3
15.2 Tools for Developing and/or Packaging Products	15-5
15.2.1 Buildmanager	15-5
15.2.2 CVS	15-5
15.2.3 Template_product	15-6
15.3 Directory Structure for a UPS Product Instance	15-6
Chapter 16: Building UPS Products	16-1
16.1 Basic Steps for Making a UPS Product	16-1
16.1.1 Build the Directory Hierarchy	16-2
16.1.2 Create the Table File	16-2
16.1.3 Declare the Product to your Development UPS Database	16-2
16.1.4 Copy the Product Executable to the bin Directory	16-3
16.1.5 Provide Product man Pages	16-3
16.1.6 Test the Product	16-4
16.2 Specifics for Different Categories of Products	16-4
16.2.1 Unflavored Scripts	16-4
16.2.2 Pre-built Binaries	16-5
16.2.3 Products Requiring Build (In-House and Third-Party)	16-6
16.2.4 Overlaid Products	16-7
16.3 Sample Auxiliary Files	16-8
16.3.1 README	16-8
16.3.2 INSTALL_NOTE	16-9
16.3.3 RELEASE_NOTES	16-9

Chapter 17: Making Products Available For Distribution	17-1
17.1 Product Distribution Overview	17-1
17.2 Creating Product Tar Files	17-2
17.3 Adding a Product	17-3
17.3.1 Product Categories Defined for KITS	17-3
17.3.2 Examples	17-4
17.4 Adding an Independent Table File	17-5
17.5 Replacing a Component (Table File or ups Directory)	17-6
17.6 Adding/Changing a Chain	17-7
17.7 Deleting a Product or Component	17-8
17.8 Cloning a Product	17-8
17.9 Including Source in one of Fermilab's CVS Repositories	17-9
17.10 Product Announcement Policies	17-10
Chapter 18: Using template_product to Build and Distribute UPS Products	18-1
18.1 Overview	18-1
18.2 Accessing template_product	18-2
18.3 Cloning template_product	18-2
18.4 The Top-Level Makefile	18-3
18.5 Inserting your Product into the Template	18-4
18.6 Building the Product	18-4
18.6.1 Add Build Instructions	18-4
18.6.2 Run the Initial Build	18-4
18.6.3 Add Build Instructions to Top-Level Makefile	18-4
18.6.4 Rebuild Instructions	18-5
18.7 Testing your Product	18-5
18.8 Customizing your Tar File	18-5
18.9 Adding your Product to a Distribution Node	18-6
18.9.1 Add Product to fnkits	18-7
18.9.2 Specify Multiple Flavors	18-7
18.10 Adding your Product Source to a CVS Repository	18-8
18.11 Removing your Product from a Distribution Node	18-8
Chapter 19: Checklist for Building and Distributing Products	19-1
19.1 Pre-build Checklist	19-1
19.2 Build the Product	19-2
19.3 Test the Product	19-2
19.4 Distribute to fnkits as "test"	19-3
19.5 Announce the Product	19-3
19.6 Distribute to fnkits as "current"	19-4

Part V: Distribution Node Maintainer's Guide

Chapter 20: Product Distribution Server Configuration	20-1
20.1 How A Server Responds to a UPD Client Command	20-1
20.1.1 The Process for upd addproduct	20-2
20.1.2 The Process for upd install	20-2
20.2 Accounts Required for Distribution Server	20-3
20.2.1 The updadmin Account	20-3
20.2.2 The ftp Account	20-3
20.2.3 The wwwadm Account	20-4
20.3 Web Server Configuration	20-5
20.3.1 The cgi Scripts Used to Access Distribution Database	20-5
20.3.2 Restricting Access to Distribution Database	20-6
20.3.3 Prerequisites for Modifying the Distribution Database	20-7
20.3.4 Permissions on Files Created in the Distribution Database	20-7
20.4 FTP Server Configuration	20-7
20.5 UPD Configuration Items	20-9
20.5.1 Archive File Keywords and \${SUFFIX}	20-9
20.5.2 Pre- and Postdeclare ACTIONS	20-10
20.6 Administrative Tasks and Utilities	20-10
20.6.1 Reporting FTP and Web Server Activity Using Ftpweblog ..	20-10
20.6.2 Restricting Access for Uploads to Distribution Database	20-11
20.6.3 Restricting Access for Downloads from Distribution Database	20-11
20.6.4 Restricting Distribution of Particular Products	20-11
20.6.5 Flagging Special Category Products Using Optionlist	20-12
20.6.6 Searching FTP Server Logfiles Using Searchlog	20-13
20.7 Product Distribution via CD-ROM	20-14
Chapter 21: Configuration of the fnkits Product Distribution Node	21-1
21.1 UPS Configuration for KITS Database	21-1
21.2 UPS Configuration for local Product Database	21-1
21.3 UPD Configuration	21-2
21.3.1 updconfig File Organization	21-2
21.3.2 The Recognized Product Categories	21-3
21.3.3 Matching Product Categories to updconfig Stanzas	21-3
21.3.4 Location and File Name Definitions	21-4
21.3.5 Pre- and Postdeclare ACTIONS	21-4
21.4 fnkits Server Maintenance	21-6
21.4.1 User Accounts and Group Ids	21-6
21.4.2 Database and Configuration File Locations	21-6
21.4.3 Web Server and FTP Log File Information	21-7

Part VI: UPS and UPD Command Reference

(Part VI is listed is listed in the front section of the table of contents.)

Chapter 22: UPS Command Reference	22-1
Chapter 23: UPD/UPP Command Reference	23-1
Chapter 24: Generic Command Option Descriptions	24-1
Chapter 25: UPS/UPD Command Usage	25-1
Chapter 26: Product Instance Matching in UPS/UPD Commands	26-1

Part VII: Administrator's Reference

Chapter 27: Information Storage Format in Database and Configuration Files	27-1
27.1 Overview of File Types	27-1
27.2 Keywords: Information Storage Format	27-2
27.2.1 What is a Keyword?	27-2
27.2.2 Keyword Syntax	27-2
27.2.3 User-Defined Keywords	27-2
27.2.4 How UPS/UPD Sets Keyword Values	27-3
27.3 Flexibility of File Syntax	27-3
27.4 List of Supported Keywords	27-3
27.5 Syntax for Assigning Keyword Values	27-8
27.6 Usage Notes on Particular Keywords	27-9
27.6.1 COMPILE_DIR, COMPILE_FILE and @COMPILE_FILE ..	27-9
27.6.2 PROD_DIR_PREFIX, PROD_DIR and @PROD_DIR	27-9
27.6.3 STATISTICS	27-9
27.6.4 TABLE_FILE and @TABLE_FILE	27-10
27.6.5 UPS_DIR and @UPS_DIR	27-11
27.6.6 _UPD_OVERLAY	27-11
Chapter 28: Version Files	28-1
28.1 About Version Files	28-1
28.2 Keywords used in Version Files	28-2
28.3 Version File Examples	28-3
28.3.1 Sample Version File for exmh v1_6_6	28-3
28.3.2 Sample version file for foo v2_0	28-4
28.4 Determination of ups Directory and Table File Locations	28-5
Chapter 29: Chain Files	29-1
29.1 About Chain Files	29-1
29.2 Keywords Used in Chain Files	29-2

29.3 Chain File Examples	29-3
29.3.1 Sample chain file for exmh v1_6_6	29-3
29.3.2 Sample chain file for foo v2_0	29-3
Chapter 30: The UPS Configuration File	30-1
30.1 dbconfig File Organization	30-1
30.2 Keywords Used in dbconfig	30-1
30.3 Sample dbconfig File	30-2
Chapter 31: The UPD Configuration File	31-1
31.1 updconfig File Organization	31-1
31.2 Product Instance Identification and Matching	31-2
31.3 Defining Locations for Product Files	31-3
31.3.1 Required Locations	31-3
31.3.2 Read-Only Variables Usable in Location Definitions	31-4
31.3.3 Sample Location Definitions	31-5
31.4 Pre- and Postdeclare Actions	31-5
31.4.1 ACTION Keyword Values	31-6
31.4.2 The execute Function	31-6
31.5 Examples	31-7
31.5.1 Generic Template updconfig File	31-7
31.5.2 Distribution from the fnkits Node Only	31-8
31.5.3 Customized Treatment of ups Directory and Table Files	31-8
31.5.4 Implementing Multiple Configurations	31-9
31.5.5 Sample Configuration for AFS Space Using ACTIONS	31-10
31.5.6 Distribution Node Configuration	31-10
Chapter 32: The UPP Subscription File	32-1
32.1 UPP Subscription File Header	32-1
32.2 Stanzas	32-2
32.2.1 Product Instance Identification	32-2
32.2.2 Conditions and Instructions	32-2
32.3 Examples	32-3
32.3.1 Sample UPP Subscription File	32-3
32.3.2 A Longer Annotated Example	32-4

Part VIII: Developer's Reference

Chapter 33: Actions and ACTION Keyword Values	33-1
33.1 Overview of Actions	33-1
33.2 UPS Command Actions	33-1
33.2.1 UPS Commands as Keyword Values	33-1
33.2.2 "Uncommands" as Keyword Values	33-2
33.3 Chain Actions	33-3
33.3.1 Chains as Keyword Values	33-3
33.3.2 "Unchains" as Keyword Values	33-3

33.4	The “Unknown Command” Handler	33-3
33.5	Actions Called by Other Actions	33-4
Chapter 34:	Functions used in Actions	34-1
34.1	Overview of Functions	34-1
34.2	Reversible Functions	34-1
34.3	Function Descriptions	34-2
34.3.1	addAlias	34-2
34.3.2	doDefaults	34-3
34.3.3	envAppend	34-3
34.3.4	envPrepend	34-4
34.3.5	envRemove	34-4
34.3.6	envSet	34-5
34.3.7	envSetIfNotSet	34-5
34.3.8	envUnset	34-5
34.3.9	exeAccess	34-6
34.3.10	exeActionOptional	34-6
34.3.11	exeActionRequired	34-6
34.3.12	execute	34-7
34.3.13	fileTest	34-7
34.3.14	pathAppend	34-8
34.3.15	pathPrepend	34-8
34.3.16	pathRemove	34-9
34.3.17	pathSet	34-9
34.3.18	prodDir	34-9
34.3.19	setupEnv	34-10
34.3.20	setupOptional	34-10
34.3.21	setupRequired	34-10
34.3.22	sourceCompileOpt	34-11
34.3.23	sourceCompileReq	34-11
34.3.24	sourceOptCheck	34-12
34.3.25	sourceOptional	34-13
34.3.26	sourceReqCheck	34-13
34.3.27	sourceRequired	34-14
34.3.28	unAlias	34-14
34.3.29	unProdDir	34-14
34.3.30	unsetupEnv	34-15
34.3.31	unsetupOptional	34-15
34.3.32	unsetupRequired	34-16
34.3.33	writeCompileScript	34-16
34.4	Functions under Consideration for Future Implementation	34-17
34.5	Examples of Functions within Actions	34-18
34.5.1	A setup Action	34-18
34.5.2	A “declare as current” Action	34-18
34.6	Local Read-Only Variables Available to Functions	34-18
34.6.1	List of Current Read-Only Variables	34-19
34.6.2	Read-Only Variables under Consideration for the Future	34-21

Chapter 35: Table Files	35-1
35.1 About Table Files	35-1
35.2 When Do You Need to Provide a Table File?	35-1
35.3 Recommendations for Creating Table Files	35-2
35.4 Table File Structure and Contents	35-2
35.4.1 Basic Structure	35-2
35.4.2 Grouping Information	35-3
35.4.3 The Order of Elements	35-3
35.5 Product Dependencies	35-4
35.5.1 Defining Dependencies	35-4
35.5.2 Product Dependency Conflicts	35-4
35.6 Table File Examples	35-6
35.6.1 Example Illustrating Use of FLAVOR=ANY	35-6
35.6.2 Example Showing Grouping	35-6
35.6.3 Example with User-Defined Keywords	35-7
35.6.4 Examples Illustrating ExeActionOpt Function	35-8
Chapter 36: Scripts You May Need to Provide with a Product	36-1
36.1 configure and unconfigure	36-1
36.2 tailor	36-3
36.3 current and uncurrent	36-3
36.4 start and stop	36-3
Chapter 37: Use of Compile Scripts in Table Files	37-1
37.1 Overview	37-1
37.2 Usage Information	37-1
Chapter 38: Creating and Formatting Man Pages	38-1
38.1 Creating the Source Document (Unformatted)	38-2
38.1.1 Source File Format	38-2
38.1.2 Man Page Information Categories	38-3
38.1.3 Example Source File	38-4
38.2 Formatting the Source File	38-5
38.2.1 nroff	38-5
38.2.2 groff	38-6
38.3 Converting your Man Page to html Format	38-6
Glossary	GLO-1
Index	IDX-1

About this Manual

This chapter provides an introduction to the *Complete Guide and Reference Manual for UPS, UPD and UPP v4*. In particular you will find:

- the overall structure, the purpose and the intended audience of the manual
- what parts of the manual you need
- where to obtain this manual and where to look for updates
- the typeface conventions and symbols used throughout the document
- an invitation to readers to send us comments

This manual is published in three submanuals: GU0014A, GU0014B, and GU0014C. The structure of the document and its division into these sections is discussed in the following sections.

1. Document Structure, Purpose and Intended Audiences

The *UPS and UPD v4 Reference Manual* is intended for several different user groups as listed on the next page. To best accommodate the different types of users, the manual is divided into five user guides (Parts I-V):

- Part I *Overview and End User's Guide*
- Part II *Product Installer's Guide*
- Part III *System Administrator's Guide*
- Part IV *Product Developer's Guide*
- Part V *Distribution Node Maintainer's Guide*

and three reference manuals (Parts VI-VIII)

- Part VI *UPS and UPD Command Reference*
- Part VII *Administrator's Reference*
- Part VIII *Developer's Reference*

The user guides explain and illustrate the **UPS/UPD/UPP** tasks associated with each user group. The reference guides provide detailed information on commands, concepts, file structure/contents, and so on. On the following page is a guide to which parts of the manual you are likely to need, according to your job functions. Notice that we recommend Parts I and VI for all users:

Parts	User Functions
A: For All Users	
Part I <i>Overview and End User's Guide</i>	<i>End Users:</i> List product information in a UPS database on a user system; Access installed software products Access FermiTools ^a software products (<i>Other user groups' functions described later in table</i>)
Part VI <i>UPS and UPD Command Reference</i>	
B: For Product Installers, UPS Database Administrators, System Administrators of User Machines, Distribution Node Maintainers	
Part II <i>Product Installer's Guide</i>	<i>Product Installers:</i> Install software products from a UPS product distribution node into a UPS database on a user system; Install products into the AFS-space UPS database
Part III <i>System Administrator's Guide</i> and Part VII <i>Administrator's Reference</i>	<i>System Administrators, UPS Database Administrators:</i> Maintain UPS products in a UPS database; Install UPS/UPD/UPP on a user system; Configure UPS on a user system; Configure UPD on a user system; Configure UPP on a user system; Configure an installed product to start/stop automatically at boottime/shutdown
Part V <i>Distribution Node Maintainer's Guide</i>	<i>Distribution Node Maintainers:</i> Install UPS/UPD on a distribution system; Configure UPS and UPD on a distribution system; Configure Web and anonymous FTP servers on a distribution system Maintain UPS database on a distribution system
C: Product Developers	
Part IV <i>Product Developer's Guide</i>	<i>Product Developers and Maintainers:</i> Develop and maintain software products that are intended to be distributed in accordance with UPS standards; Adapt pre-existing or third-party software to conform to UPS standards; Distribute products
Part VIII <i>Developer's Reference</i>	

a. Fermilab-written software products that are made publicly available.



The table above lists rather generally the topics that the manual covers. Note that it is *not* the purpose of this document to provide information on:

- general UNIX system administration
- general UNIX or Fermilab information (see instead *UNIX at Fermilab*, GU0001)
- the use of any particular software product other than **UPS/UPD/UPP**

CDF and D0 collaborators: Also see *A UNIX Based Software Management System* (GU0013) at http://www-cdf.fnal.gov/offline/code_management/run2_cmgt/run2_cmgt.html to find information describing how **UPS** and **UPD** have been implemented in your experiments' code management systems.

2. Availability

Copies of the *UPS and UPD v4 Reference Manual* (GU0014A, B, and C), can be obtained from the following sources:

Web

<http://www.fnal.gov/docs/products/ups/ReferenceManual/>

This can be accessed under **Documentation** on the Computing Division home page. Search using any of the following keywords: afs, develop(ment), distribute(tion), fermitools, GU0014, install(ation), kits, maintain(enance), man page, product, system administration, unix, upd, upp, ups

Paper Copies

Wilson Hall, 8th floor, NE (just across from what used to be the Computing Division library)

3. Updates

Pending subsequent releases of this manual, updates will be maintained on the Web with the on-line version of the manual. To get there from the Computing Division home page, select **Documentation**, request *GU0014* and follow the pointers (see "Web" under section 2. *Availability*).

4. Conventions

The following notational conventions are used in this document:

- | | |
|---------------|--|
| bold | Used for product names (e.g., UPS). |
| <i>italic</i> | Used to emphasize a word or concept in the text. Also used to indicate logon ids and node names. |
| typewriter | Used for filenames, pathnames, contents of files, output of commands. |

typewriter-bold	Used to indicate commands and prompts.
[...]	In commands, square brackets indicate optional command arguments and options.
	When shown in a command example (e.g., x y z), separates a series of options from which one may or must be chosen (depends if enclosed in square brackets). In UNIX commands, used to pipe output of preceding command to the following one.
' ... '	Single vertical quotes indicate apostrophes in commands.
" ... "	Double vertical quotes indicate double quotes in commands
...	In a command, means that a repetition of the preceding parameter or argument is allowed.
%	Prompt for C shell family commands (% is also used throughout this document when a command works for both shell families).
\$	Prompt for Bourne shell family commands; also standard UNIX prefix for environment variables (e.g., \$VAR means “the value to which VAR is set”).
\	UNIX standard quoting character; used in commands throughout the manual to indicate that the command continues to the next line
<...>	In commands, variables, pathnames and filenames, angle brackets indicate strings for which reader must make a context-appropriate substitution. For example, \$<PRODUCT>_DIR becomes \$EMACS_DIR for the product emacs .
{ }	In local read-only variables, e.g., \${UPS_PROD_DIR}, string should be used as shown with the {}.

All command examples are followed by an implicit carriage return key.

Some of the files discussed in this document are shell family-specific, and thus come in pairs. Their filenames carry the extensions `.sh` and `.csh`. We often refer to a pair of these files as `filename.[c]sh`.

The following symbols are used throughout this document to draw your attention to specific items in the text:



A “bomb”; this refers to something important you need to know in order to avoid a pitfall.



This symbol is intended to draw your attention to a useful hint.

5. Your Comments are Welcome!

The *UPS and UPD v4 Reference Manual* may contain some errors, however we endeavor to minimize the error count! We encourage all the readers of this document to report back to us:

- errors or inconsistencies that we have overlooked
- any parts of the manual that are confusing or unhelpful -- please offer *constructive* suggestions!
- other topics to include (keeping in mind the purpose of the manual)
- tricks, hints or ideas that other users might find helpful

Send your comments via email to cdlibrary@fnal.gov.

Part I Overview and End User's Guide

Chapter 1: *Overview of UPS, UPD and UPP v4*

UPS, UPD and **UPP** are the utilities provided by Fermilab's Computing Division for managing and standardizing software product development, distribution, support and access. This overview chapter describes these utilities briefly, and discusses the reasons for which this particular product methodology was chosen and developed. The chapter also describes:

- features of products maintained and distributed under this system (called **UPS** products)
- **UPS** databases
- the **UPS** software environment

Chapter 2: *UPS Operations for the End User*

This chapter describes how to get information about **UPS** products that are installed on a user system and declared to a **UPS** database, and how to access them.

Chapter 1: Overview of UPS, UPD and UPP v4

UPS, **UPD** and **UPP** are the utilities provided by Fermilab's Computing Division for managing and standardizing software product development, distribution, support and access. This overview chapter describes these utilities briefly, and discusses the reasons for which this particular product methodology was chosen and developed. The chapter also describes:

- features of products maintained and distributed under this system (called **UPS** products)
- **UPS** databases
- the **UPS** software environment

1.1 Introduction to UPS, UPD and UPP

UPS (UNIX Product Support) is a software support toolkit developed at Fermilab for the management of software products on local systems by the system administrators and users. It was also designed to facilitate the product distribution and configuration management tasks of the product providers. The three principal user benefits are:

- a uniform interface for accessing all products on a UNIX (or UNIX-like, e.g., Cygwin) system via the **setup** command
- unified and coordinated support of in-house and vendor-supplied software across all the supported UNIX operating systems
- the capacity for running multiple concurrent versions of products on the same system, with a standard, simple version-selection mechanism

UPD (UNIX Product Distribution) is a companion product to **UPS**, and provides the functionality for uploading/downloading products between local systems and product distribution servers.

UPP (UNIX Product Poll) is a layer on top of **UPD** that allows a client to request notification of changes in a distribution node database and to download pre-specified products. **UPP** can be automated. This is a useful tool for keeping abreast of changes/enhancements to your favorite products.

A **UPS/UPD/UPP** installation usually has several parts:

- one or more databases; a **UPS** database is a directory which functions as a central repository of information about products and contains pointers to products.
- a **UPS** database configuration file which contains system-specific information that customizes the **UPS** installation on a node or cluster
- a **UPD** configuration file that tells **UPD** which database to use to declare a product, in what directories to unwind the product and its related files, what naming conventions to use for various metadata files, and other information.

- the **UPS**, **UPD** and **UPP** executables which manipulate and view the database(s) on client and server nodes
- a **UPP** subscription file that lists product instances you are interested in tracking and a list of commands to perform when new instances appear (notify by email, install the instance, etc.).
- an entry in the crontab file to run **UPP** at some periodic interval.

1.2 Motivation for the UPS Methodology

Why has Fermilab developed and implemented the **UPS** product support methodology? The principal reason is to provide self-contained products. Each release of a self-contained product can be installed and accessed independently of other releases. There are two main advantages to this, which are especially important for applications such as real-time data acquisition:

- Multiple versions of a software product can be made available concurrently. This is useful in many situations, especially when some products depend on particular versions of others for compiling or running. Multiple concurrent versions also makes possible the second advantage:
- You can back out of a new software release **completely** and **assuredly** if something goes wrong, and immediately start up a previous tried-and-true release.

The **UPS** methodology also provides tracking. A glance at the **UPS** database tells you what version of a product you're running; you don't need to keep track of it elsewhere or risk forgetting. You can also easily tell if different machines are running the same version of a product.

All users of UNIX utilities and software products on a system running **UPS** will appreciate these additional features:

- capability for supporting a cluster of UNIX systems from different vendors fitted with common software products (and optionally a common products disk)
- an easy and consistent method of accessing a variety of software products
- wider availability of supported software
- linking of dependent products in such a way that when a product is made accessible for use, any products that it relies upon for proper functioning are also automatically made accessible

Further advantages that product installers, system administrators and product providers will encounter include:

- a rich set of commands for installing and maintaining products
- a single product development/distribution methodology and set of standards
- a wider audience available to test new products

1.3 UPS Products

Products distributed and managed by **UPS** on a distribution or user node are called *UPS products*. Typically, **UPS** products on a system are declared in a **UPS** database on that system. **UPS** products can be maintained on different disks, and/or in different directories. Each product in a **UPS** database is managed via a set of files that provide product management information to **UPS**, e.g., the location of each copy of the product, what its status is (e.g., appropriate for general use, for testing, for development), what needs to be done when the product is installed or accessed, and so on.

UPS products are generally self-contained and portable, laid out in a directory structure under what we call a *product root directory*. The structure of a product's directory tree is not dictated by **UPS**, but generally it includes (at least) areas for the executables (`bin`), for test scripts (`test`), and for documentation (e.g., `man`, `catman`, `docs`, `html`).

1.3.1 Versions

UPS supports multiple concurrent versions of software products. Each version of a product is installed and accessed independently of other versions. When a new version of a product becomes available, an existing directory tree is not replaced with another; rather, a branch is added for the new version. (See the diagram in section 1.4.2 *UPS Database Structure*.)

1.3.2 Flavors

Many programs require separate compilations for each of the different UNIX operating systems. **UPS** allows you to maintain a separate directory tree for each compilation (and related files) of the same product. To distinguish between different OS-dependent compilations, we use the term *flavor*. This additional term allows us to maintain the same product name and version across the different operating systems and OS versions, which is desirable since the same program source files are generally used in the separate compilations.

Several different copies of a product may exist on a given system. When you use **UPS** commands to manipulate or use a product, the system needs to have enough information to select the appropriate one. All **UPS** commands support a `-f` option¹ allowing you to specify flavor.

The flavor of a particular product compilation as declared in a **UPS** distribution database may or may not indicate the version of the OS for which it was compiled. The standard we have adopted for flavors in the `KITS` distribution database on the Computing Division's central product distribution node `fnkits.fnal.gov` is:

- For products which have no compiled programs, and are thus operating system-independent, a special flavor of `NULL` is used.
- Flavored products are declared with the full flavor specification of the OS on which they are built.

1. There are also the number options `-0` through `-3` which can be used in place of `-f`; `-H` can be used to run a **UPD** command as if the local flavor were as specified. See Chapter 24: *Generic Command Option Descriptions* for descriptions.

On user systems the flavor of a product may be declared differently. For example, products which can run on multiple releases of a single OS are sometimes declared on user systems with the UNIX OS name only. For example, a product that runs on all IRIX releases but is declared in `KITS` as `IRIX+6.5` may be declared on a user system with the flavor `IRIX`. (This practice is declining. It is discouraged because it is difficult to maintain and can cause setup problems when the product is a dependent product of another. Dependent products are defined in section 1.3.6 *Product Dependencies*.)

1.3.3 Qualifiers

The product developer may include information about options used at compilation time (e.g., `debug` or `optimized`) or other qualifying information for easy identification of special compilations. This information is declared in the form of *qualifiers* in a distribution database. When a product is declared to **UPS**, the installer has the choice of declaring these qualifiers or omitting them in the declaration.

Declaring the qualifiers allows full identification of the product compilation on your system. The drawback is that to access a product compilation declared with one or more qualifiers, a user *must* specify the qualifier(s) when accessing that compilation.



No standard set of qualifiers has been defined; the naming of qualifiers is at the discretion of the product provider, and thus may vary from product to product.

1.3.4 Product Instances

Each installed copy of a product that is declared to a **UPS** database is called an *instance* of the product. Within a database, a product instance is distinguished by a unique combination of product name, version, flavor, and qualifiers. In the case of multiple databases, the database specification is often also needed to distinguish an instance because identical instances can be maintained in different databases. The concept of *chains*, discussed in section 1.3.5, allows users to easily access the appropriate instance of a product according to their needs without having to remember its version number and other details.

1.3.5 Chains

We mentioned earlier that **UPS** supports multiple versions of software products on a machine. End users do not find it convenient to specify product version numbers each time they setup a product. This is especially true if product setups are needed at login. Most users want to run the latest, tested, approved version of products without having to keep track of the version numbers.

To allow users to specify the version of a product according to its *status* rather than by its version number, **UPS** supports *chains* to product versions. A chain can be thought of as an alias for a particular product instance. It indirectly “attaches” a chain name to a product instance, thereby tagging the product instance according to its status.

Five standard chains, have been defined for use: `current`, `new`, `test`, `old`, and `development`.

Chain Names, Options, and Usage

Chain	Option	Usage
current	-c	default instance recommended for general use
new	-n	tested instance that is not yet current
test	-t	instance installed for testing
development	-d	instance under development
old	-o	older instance that was previously current

Additional chains may be defined by developers and installers and other users. The command option **-g** is provided for this purpose. For an example of how it's used, see section 10.2 *Declare a Chain*.

1.3.6 Product Dependencies

Many **UPS** products require that other **UPS** products be installed, declared and setup for proper functioning or for use of special features. These other products are referred to as *dependencies*. A dependency is generally an independent product that is maintained in its own individual product root directory. Non-**UPS** products are sometimes declared as dependencies as well; for instance, **gcc** is not installed under **UPS** on some machines, but is a dependency of some **UPS** products. End users don't normally need to know what dependencies a particular product has, as long as the product runs without problems.

We distinguish two categories of dependencies: those which are required for the main product to function, and optional ones which generally enable nonessential features of the main product.

The coupling of products with their dependencies facilitates product setup (setup is described in section 2.4 *Setting up a Product*). You need only setup a single **UPS** product to access any and all of the products listed as dependencies for that product.

Multiple levels of dependencies are possible. As an example, the mail product **exmh** has several "first level" dependencies, one of which is **www**. **www** has dependencies of its own, which in turn may have dependencies, and so on. These are all referred to as "lower level" dependencies of **exmh**.

UPS product dependencies can exist across databases. For this to work, the databases in which they are declared must all be included in \$PRODUCTS, or in a database list specified on the command line.

A note about what used to be called *build dependencies*: These don't exist as such as of **UPS** v4. The qualifier "BUILD" or "build" now provides essentially the same function as the old build dependencies. There is an example in the reference section 22.1 *setup* that illustrates the use of this qualifier.

1.3.7 Product Overlays

An additional class of required product is supported by **UPD**, called an *overlaid product*. An overlaid product gets distributed and maintained in the product root directory of its main product. For example, the overlaid products **cern_bin**, **cern_ups**, **cern_lib**, etc., all reside in the product root directory for the main product **cern**. A patch is another good example of the use of overlaid products. The set of products overlaid on a main product is collectively referred to as *the overlay*. An overlaid product is not precluded from being a dependency of other **UPS** products.

1.4 UPS Database Overview

The information **UPS** needs for managing products is maintained in a database. A **UPS** database is a directory tree which contains a subdirectory for each product declared to the database, the subdirectory having the same name as the product. It also usually contains the hidden subdirectories `.upsfiles` with **UPS** database configuration information, and `.updfiles` with **UPD** configuration information.

Within each product-specific directory under the **UPS** database directory, a set of ASCII files collectively contains the **UPS** management information for the product on that system. We call these files *UPS database files*.

UPS commands refer to the database directory via the **UPS** environment variable `$PRODUCTS` described in section 1.7.2, or the `-z <databaseList>` option. `$PRODUCTS` can be set to point to one or to several directories, thus allowing support for multiple databases. This allows users to maintain one or more private databases in addition to or instead of the common one(s).

1.4.1 UPS Database Files

There are two types of **UPS** database files for each product: version and chain files.

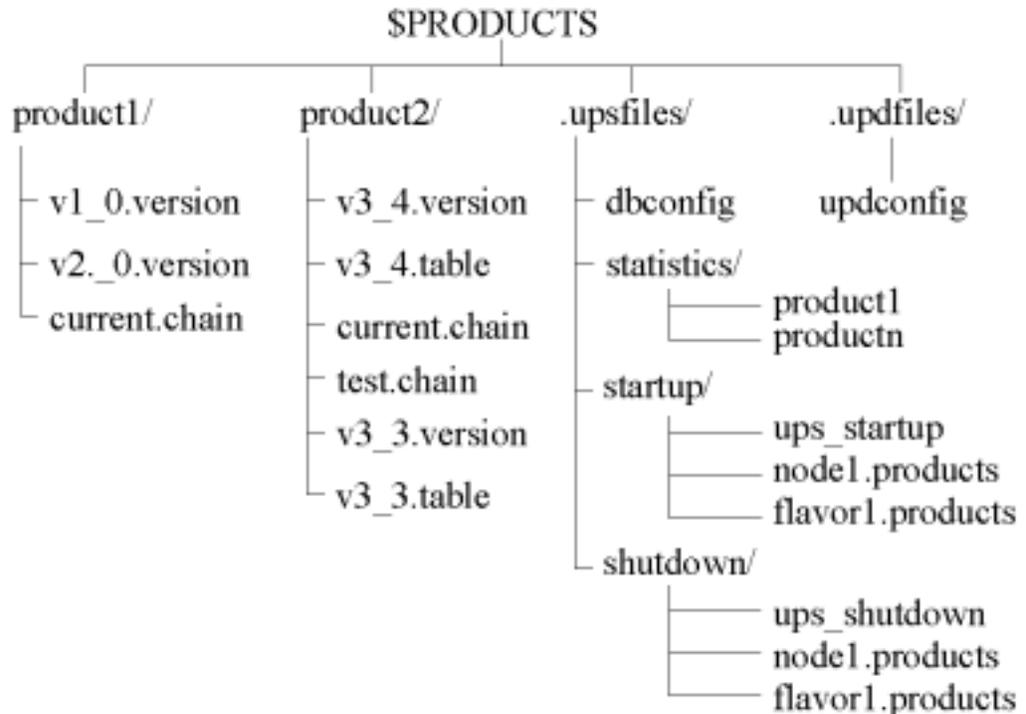
- *Version files* tell **UPS** where to find all the files associated with a particular version of a product on the local system, and contain some other system-specific information. They are generally named according to the scheme `vx_y.version`, e.g., `v1_0.version`. These are described in Chapter 28: *Version Files*.
- *Chain files* are optional and contain pointers to version files, thus providing convenient access to particular product versions on the local system. They are generally named according to the scheme `chainname.chain`, e.g., `current.chain`. These are described in Chapter 29: *Chain Files*.

Another file that strictly speaking is not a database file, but is also used in product management is a *table file*. Table files contain information which is independent of any local installation and which is specific to one or more particular instances of the product. For example, table files tell **UPS** what needs to be done to configure a product or to make it accessible for use. Table files are provided by the product developer when needed. Not all products have table files. And conversely, some products consist only of a table file. Table files are described in Chapter 35: *Table Files*.



Whereas version and chain files are constrained to reside under the **UPS** database, table files can reside anywhere. Table files are usually kept either in the database or within the product instance's root directory structure. Table file paths are indicated in version files, and they are thereby accessible to **UPS** regardless of location.

1.4.2 UPS Database Structure



1.5 Using UPS Without a Database

It bears mentioning that **UPS** can be installed on a machine to manage products without a **UPS** database, albeit in a limited way. (Except where noted, this manual is written with the assumption that **UPS** is used with a database.) In the absence of a database, there are no database files and every **UPS/UPD** command must include all of the information that normally would have been read from a database. In particular, all commands require specification of the table file name, and usually its location. Functionality requiring or operating on a **UPS** database is not supported when **UPS** is implemented in this way.

This flexibility is provided primarily for off-site users who for one reason or another do not want to maintain a **UPS** database on their local system. Product developers may also work in this type of environment, especially if they're using **CVS** or another code management product; this is described in section 15.2.2 *CVS* under 15.2 *Tools for Developing and/or Packaging Products*.

1.6 UPS and UPD Commands

Please see Part VI *UPS and UPD Command Reference* of this manual for complete information on **UPS** and **UPD** commands. To get you started, we describe briefly here the command syntax and defaults used.

1.6.1 Syntax

Most **UPS** and **UPD** commands are of the form **ups <command>** or **upd <command>** (the exceptions are **setup** and **unsetup**), and take a variety of command line options and arguments. Multiple arguments must be separated by colons (:). The standard syntax is:

```
% ups <command> [<options>] <product> [<version>]
```

For example:

```
% ups list -f IRIX+6:OSF1+V3 xemacs v20_4
```

The first occurring unflagged argument on the line after the command is generally interpreted as the product name¹, and the next (if any) is interpreted as the version. With that limitation, the name, version and options can occur anywhere on the command line.

1.6.2 Defaults

If no database is given, **UPS** uses \$PRODUCTS to determine the database. If no instance-identifying options or version are given, **UPS/UPD** operates on the instance declared as current for the flavor of the machine on which the command is issued (to the highest specification level possible). If there is no instance declared as current or if the current instance has any qualifiers, then the default instance matching will fail.

For **UPD** commands, if no product distribution node is specified, **UPD** uses the central Computing Division product server *fnkits.fnal.gov* as the default. (The product distribution database on this node is known as `KITS`.)

1. An exception is the `<componentList>` element in the `upd update` command, documented in section 23.12 *upd update*.

1.7 The UPS Environment

1.7.1 Initializing the UPS Environment

In order to access and use **UPS** products or manipulate a **UPS** database, your environment must be initialized to make the **UPS** commands available to you.

Systems using Fermi UNIX Environment

Many on-site Fermilab systems, and some off-site nodes as well, have been configured with the Fermi UNIX Environment (FUE).¹ If your system runs FUE, your **UPS** environment gets initialized automatically when you log in.



The **UPS** initialization does not carry over to any scripts which either create a new login process or invoke a shell from the “other” shell family. If such a script needs to setup and run **UPS** products from that new process, you need to include a line in the script which sources the appropriate `setups.[c]sh` file, as described below.

Systems NOT using Fermi UNIX Environment

If your system does not run FUE, you will need to initialize your **UPS** environment yourself unless your system administrator has taken care of this. The **UPS** initialization is accomplished by sourcing a **UPS**-provided script; namely, one of the following (depending on your login shell):

Bourne shell family	<code>setups.sh</code>
C shell family	<code>setups.csh</code>

It must be sourced from the directory where the **UPS** setup files have been installed on your system. Virtually all supported systems on-site provide “courtesy links” in `/usr/local/etc` so that you don’t need to know exactly where the `setups.[c]sh` files actually reside². If the `setups.[c]sh` files are *not* in `/usr/local/etc` and your system doesn’t maintain these courtesy links, you will need to ask your system administrator or **UPS** database maintainer where to find these files.

You can either source the `setups.[c]sh` scripts manually (for occasional use), or you can add the following to your `.[c]shrc` file so that **UPS** gets initialized every time you log in:

Bourne shell family	<code>. /path/to/setups.sh</code>
C shell family	<code>source /path/to/setups.csh</code>

You also need to include this line in any other scripts which setup and invoke **UPS** products, once per login session and any time a shell from the other shell family is invoked.

1. FUE was updated in the fall of 1999. It requires **UPS/UPD** v4_0 or higher and **systemtools** v6_0 or higher plus dependencies.

2. There are some exceptions; including the system where **UPS** development takes place.

1.7.2 Changes UPS Makes to your Environment

The following environment variables get set/modified by `setups.[c]sh`:

<code>\$PRODUCTS</code>	If only one UPS database is defined, this points to it; if two or more are defined, this variable can be set as the colon-separated ¹ list of UPS databases. The order of the databases in this list reflects the order of precedence for accessing products.
<code>\$UPS_DIR</code>	This points to the top level directory (called the product root directory) of the active instance of UPS .
<code>\$UPS_SHELL</code>	Set to sh or csh , depending on shell family in use.
<code>\$PATH</code>	Modified to include <code>\$UPS_DIR/bin</code> .
<code>\$SETUP_UPS</code>	a string containing all the information that the unsetup command needs in order to identify the active instance of UPS (e.g., <code>ups v4_5_2 -f SunOS+5 -z /path/to/db</code>) (e.g., <code>ups v4_5_2 -f SunOS+5 -z /path/to/db</code>)In addition, three aliases get defined: <code>ups</code> ² , <code>setup</code> and <code>unsetup</code> .

1. Using whitespace as a separator in place of colons is allowed in `$PRODUCTS` for backwards compatibility. However, colons are recommended. Using colons is more consistent with the new command format that requires multiple option arguments to be separated on the command line with colons.

2. The `ups` alias exists for backwards compatibility in a mixed **UPS** v3/v4 environment and will be dropped in the **UPS** release following v4_5_1.

Chapter 2: UPS Operations for the End User

This chapter describes how to get information about **UPS** products that are installed on a user system and declared to a **UPS** database, and how to access them. Since the target audience includes all levels of **UPS** users, we've tried to keep the information and examples here relatively uncomplicated. We encourage you to refer to Part VI of this manual, the *Command Reference*, for full command documentation and more sophisticated examples. In particular, see Chapter 22: *UPS Command Reference* for command descriptions and command-specific option descriptions, and Chapter 25: *UPS/UPD Command Usage* for a complete description of command syntax.

We've chosen to organize the chapter logically, first showing you how to find information about your machine and the available products, and then showing you how to access the products. The topics covered include:

- determining the flavor of your machine (**ups flavor**)
- finding what products are installed in the **UPS** database on your system, and finding information about them (**ups list**). We document the new feature **ups list -K** which provides output in script-readable format.
- determining what a product's dependencies are (**ups depend**)
- accessing (running **setup** on) a product
- removing product accessibility (running **unsetup** on a product)



To get on-line usage information or help, use the following resources:

- Run the command with `"-?"`, e.g., **ups list "-?"**, to get an option listing. The double quotes are necessary for C shell users; `-?` is interpreted by **sh**.
- Man pages are also provided; use an underscore with the **UPS** command when running **man**, e.g., **man ups_list**.

2.1 Determining your Machine's Flavor

The **ups flavor** command returns flavor information about the machine issuing the command, or for a flavor requested via the `-H <flavor>` option. The full command description for **ups flavor** is given in the reference section 22.8 *ups flavor*. When entered with no options, the command returns the full OS specification of the machine, for example:

```
% ups flavor
```

```
SunOS+5.6
```

When entered with the `-1` (long) option, it returns what we call a *flavor table*, which takes the basic flavor (e.g., SunOS) and lists it at every level of specificity that you could use to find or declare a product instance. For example:

```
% ups flavor -1
```

```
SunOS+5.6  
SunOS+5  
SunOS  
NULL  
ANY
```

You can specify a particular level using the number options: `-0`, `-1`, `-2`, `-3`, with `-3` being the most highly specified; for example:

```
% ups flavor -3
```

```
SunOS+5.6
```

```
% ups flavor -2
```

```
SunOS+5
```

```
% ups flavor -1
```

```
SunOS
```

```
% ups flavor -0
```

```
NULL
```

2.2 Listing Product Information in a Database

The `ups list` command returns information about the declared product instances in a **UPS** database. End users typically use it for finding out what products are in the database, what the current version of a product is for their machine's flavor and what other versions might be available. Product installers and other administrative users can use it to get detailed information about a product's installation and to find product files. Two output styles are provided: a formatted one that is easy for users to read, and a condensed one for parsing by a subsequent command or a script.

You specify the information you want contained in the output by including various options and arguments in the command. As is standard in **UPS**, if no chain, version or flavor is specified, and `-a` (for *all* instances) is *not* specified, **UPS** returns only the instance declared as current for the best-matched flavor of the requesting machine. Here we show the command syntax including several of the commonly-used options:

```
% ups list [-a] [-f <flavorList>] [<chainFlag>] \  
[-K <keywordList>] [-l] [-q <qualifierList>] [<product>] \  
[<version>]
```

The full command description for `ups list` is given in the reference section 22.11 *ups list*.

2.2.1 Formatted Output Style

One output style is for visual parsing (this is the default output, which we call *formatted*), with multi-line output, e.g.,:

```
% ups list xemacs
```

```
    DATABASE=/usr/upsII/ups_database/declared/oss
    Product=xemacs  Version=v19_14  Flavor=SunOS+5
    Qualifiers=""   Chain=current
```

Notice that the product name, version, flavor, qualifiers and chain(s) are the fields that get returned by default. The database (first line of the output) is included as a header, not as part of the per-instance data.

2.2.2 Condensed Output Style

The other output format is a script-readable, or *condensed*, format, provided to allow parsing by a subsequent command. Use the **-K** option (upper case) to request output in the condensed format. The **-K** option requires an argument list specifying which fields to include in the output. For guidance on parsing the condensed output in **perl** or in a **sh** script, see the reference section 22.11 *ups list*.

-K Usage Notes

- The arguments for **-K** are not case-sensitive.
- The plus sign (+) argument, e.g., **-K+**, is a shorthand for requesting the default fields **product:version:flavor:qualifiers:chain**.
- Leaving a space immediately after the **-K** is optional.
- Separate the keywords using colons (:)

Commonly Used Keyword Arguments

Some common keyword arguments used with the **-K** option are:

PRODUCT	product name
FLAVOR	product flavor
VERSION	product version
QUALIFIERS	additional instance specification information often used to indicate compilation options used by developer
CHAIN	product instance chain
+	all of the above
DATABASE (or DB)	the UPS database path; useful if there are multiple databases
DECLARER	userid of person who declared the instance
DECLARED	date/time that product instance was declared

MODIFIER userid of person who modified/updated the instance

MODIFIED date/time that product instance was modified/updated

The full list of keywords and their definitions can be found in section 22.11.4 *More Detailed Description* under the reference section 22.11 *ups list*. Administrative users may find the following three “shorthand” keywords useful:

@PROD_DIR entire path for the directory where the product is installed (usually equivalent to PROD_DIR_PREFIX/PROD_DIR)

@TABLE_FILE entire path for the table file (table file locations are described in section 28.4 *Determination of ups Directory and Table File Locations*)

@UPS_DIR product’s ups directory path; if it is not an absolute path, then it is taken relative to @PROD_DIR

2.2.3 Examples

For many of the examples that follow, the output is edited for brevity.

List All Current Products

The simplest way to request a listing of all the current product instances in your default **UPS** database is to use the **ups list** command with no options or arguments:

% ups list

```
DATABASE=/afs/fnal.gov/ups/db
  Product=admintools  Version=v1      Flavor=SunOS+5
                    Qualifiers=""   Chain=current

  Product=afsemu     Version=v1_2  Flavor=NULL
                    Qualifiers=""   Chain=current

  Product=aixserv    Version=v2_6_8  Flavor=NULL
                    Qualifiers=""   Chain=current

  Product=ansys      Version=v5_3   Flavor=SunOS+5
                    Qualifiers=""   Chain=current

  ...

  Product=ximagetools  Version=v3_1   Flavor=SunOS+5
                    Qualifiers=""   Chain=current

  Product=xntp        Version=v3_4   Flavor=SunOS+5
                    Qualifiers=""   Chain=current

  Product=xpdf        Version=v0_7   Flavor=SunOS+5
                    Qualifiers=""   Chain=current

  Product=zephyr      Version=v2_0_4  Flavor=SunOS+5
                    Qualifiers=""   Chain=current
```

Use of the **-K** option with the **+** argument provides the same information as the previous example, but condensed to one line per product, e.g.,:

```
% ups list -K+
```

```
"admintools" "v1" "SunOS+5" "" "current"
"afsemu" "v1_2" "NULL" "" "current"
"aixserv" "v2_6_8" "NULL" "" "current"
"ansys" "v5_3" "SunOS+5" "" "current"
...
"ximagetools" "v3_1" "SunOS+5" "" "current"
"xntp" "v3_4" "SunOS+5" "" "current"
"xpdf" "v0_7" "SunOS+5" "" "current"
"zephyr" "v2_0_4" "SunOS+5" "" "current"
```



If \$PRODUCTS contains multiple databases, output is returned for the selected products in all of them. However, when using **-K** the database is identified for each output line *only if* the keyword DATABASE or DB is included in the argument string (e.g., **-K+ :DB** requests the standard output fields followed by the database path):

```
% ups list -aK+:DB
```

```
"admintools" "v1" "SunOS+5" "" "current" "/afs/fnal.gov/ups/db"
"afsemu" "v1_2" "NULL" "" "current" "/usr/products/ups_database/main"
...
```

List All Product Instances

Use the **-a** option to list all instances (**-a** for all) of the product **emacs**:

```
% ups list -a emacs
```

```
DATABASE=/afs/fnal.gov/ups/db
  Product=emacs  Version=v19_30a  Flavor=IRIX+5
    Qualifiers=""  Chain=""

  Product=emacs  Version=v19_30a  Flavor=SunOS+5
    Qualifiers=""  Chain=""
...

  Product=emacs  Version=v19_34b  Flavor=SunOS+5
    Qualifiers=""  Chain=current
```

List the same information as the previous example in condensed format:

```
% ups list -aK+ emacs
```

```
"emacs" "v19_30a" "IRIX+5" "" ""
"emacs" "v19_30a" "SunOS+5" "" ""
...
"emacs" "v19_34b" "SunOS+5" "" "current"
```

Taking this example one step further, you can pipe the output to the **grep** command, for example:

```
% ups list -aK+ emacs | grep SunOS+5
```

```
"emacs" "v19_30a" "SunOS+5" "" ""
...
"emacs" "v19_34b" "SunOS+5" "" "current"
```

Specify Output Fields

Instead of using the `+ argument` with `-K` to get the default fields, you can specify particular fields. For example, to output a list of product names and version numbers for all the current products, use the command:

```
% ups list -K product:version
```

This produces:

```
"admintools" "v1"  
"afsemu" "v1_2"  
"aixserv" "v2_6_8"  
"ansys" "v5_3"  
...  
"ximagetools" "v3_1"  
"xntp" "v3_4"  
"xpdf" "v0_7"  
"zephyr" "v2_0_4"
```

Request Detailed Information for a Product Instance

Administrative users may need detailed information about a product which the `-l` option (lower case `-L`) provides. A request for information on the current instance of a single product using the long output format (not available with the `-K` option) gives the following:

```
% ups list exmh -l
```

```
DATABASE=/afs/fnal.gov/ups/db  
Product=exmh Version=v2_0_2 Flavor=NULL  
Qualifiers= " " Chain=current  
Declared="1998-11-17 15.04.41 GMT:1998-10-22 16.31.12 GMT"  
Declarer="lauri:lauri"  
Modified="1998-11-17 15.04.41 GMT:1998-10-22 16.31.12 GMT"  
Modifier="lauri:lauri"  
Home=/afs/fnal.gov/ups/exmh/v2_0_2/NULL  
No Compile Directive  
Authorized, Nodes=*  
UPS_Dir="ups"  
Table_Dir=""  
Table_File="v2_0_2.table"  
Archive_File=""  
Description=""  
Action=setup  
    prodDir()  
    setupEnv()  
    setupRequired(expect)  
    setupRequired(mh)  
    setupRequired(mimetools)  
    setupOptional(glimpse)  
    setupOptional(www)  
    setupOptional(netscape)  
    setupOptional(ispell)  
    pathPrepend(PATH, ${UPS_PROD_DIR}/bin)  
    envSetIfNotSet(NETSCAPE_DIR, "$HOME")  
Action=current  
    prodDir()  
    execute(${UPS_UPS_DIR}/current, UPS_ENV)
```

This gives a fairly long list. It is often more convenient to use the `-K` option with a list of keywords for the specific fields you need.

2.3 Finding a Product's Dependencies

The **ups depend** command lists dependencies of specified product instance(s) as declared in the local database. You can use it to determine what products will get setup along with your main product, or to determine what products need to be available in order to successfully setup the main one. Shown with some commonly-used options, the command syntax is:

```
% ups depend [-f <flavorList>] [<chainFlag>] [-j] \  
[-K <keywordList>] [-q <qualifierList>] [-R] \  
<product> [<version>]
```

The full command description for **ups depend** is given in the reference section 22.6 *ups depend*.

Dependency information is also included in the output of the **ups list -l** command. When using **ups list -l**, you must infer the dependencies by looking at the **setupRequired** and **setupOptional** functions under the SETUP action (see the example on previous page). It does not provide information on lower level dependencies.

Examples

The first example requests information for the default instance of **exmh**, which is the one chained to “current” for the best-match flavor of the machine (SunOS+5 for this example):

```
% ups depend exmh  
  
exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current  
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| |__tk v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db  
| | |__tcl v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db  
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| |__mailtools v2_3 -f NULL -z /afs/fnal.gov/ups/db -g current  
|__mimetools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
|__glimpse v3_0a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
|__www v3_0 -f NULL -z /afs/fnal.gov/ups/db -g current  
| |__netscape v4_05 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| | |__ghostview v4_0 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| | | |__ximagetools v4_0 -f NULL -z /afs/fnal.gov/ups/db -g current  
| | | | |__imagemagick v4_04 -f SunOS+5 -z /afs/fnal.gov/ups/db  
| | | | |__xfig v3_20 -f SunOS+5 -z /afs/fnal.gov/ups/db  
| | | | |__xanim v2_70_64 -f SunOS+5 -z /afs/fnal.gov/ups/db  
| | | |__xpdf v0_7 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| |__lynx v2_8_1 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
|__ispell v3_1b -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
```

We use the **-R** option to request only the dependencies listed as “required” for the same product instance as in the previous example:

```
% ups depend -R exmh  
  
exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current  
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| |__tk v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db  
| | |__tcl v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db  
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current  
| |__mailtools v2_3 -f NULL -z /afs/fnal.gov/ups/db -g current  
|__mimetools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
```

The **ups depend** command accepts the **-K** option described in section 2.2.2 *Condensed Output Style* to return a subset of the output fields. Here, we use the **-R** option again with the same product instance (this allows you to compare the output to the previous example), and we use **-K** to specify output fields:

```
% ups depend -RK product:version:flavor exmh
```

```
"exmh" "v2_0_2" "NULL"
"expect" "v5_25" "SunOS+5"
"tk" "v8_0_2" "SunOS+5"
"tcl" "v8_0_2" "SunOS+5"
"mh" "v6_8_3c" "SunOS+5"
"mailtools" "v2_3" "NULL"
"mimertools" "v2_7a" "SunOS+5"
```

Use the **-j** option to request “just” the direct dependencies of the specified product, omitting dependencies of dependencies (again, the same instance is used to allow comparison):

```
% ups depend -j exmh
```

```
exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__mimertools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__glimpse v3_0a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__www v3_0 -f NULL -z /afs/fnal.gov/ups/db -g current
|__netscape v4_05 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__ispell v3_1b -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
```

2.4 Setting up a Product

An installed, declared **UPS** product instance requires that the **setup** command be issued prior to use, unless it has already been set up as a dependency of another product. The **setup** command performs the necessary operations in your login environment to make an installed, declared product accessible to you. Typically, the operations include modifying environment variables or adding to your **\$PATH**. Many options are available for the **setup** command, as listed in the reference section 22.1 *setup*. Here we show the syntax with a few of the more commonly-used options:

```
% setup [-f <flavor>] [<chainFlag>] [-q <qualifierList>] \
[-z <databaseList>] <product> [<version>]
```



Note that you can only have one instance of a product setup at a time. Each time you run **setup** on an additional instance of the same product, the previously active instance is automatically unset up first.

2.4.1 The setup Command for the Typical Case

Current Instance

Typically users want to setup the default current version of a product. This is very simple, you just need to specify the product name. For example, to setup the current instance of **tex** for the best-matched flavor of your OS, enter:

```
% setup tex
```

Other Chained Instance

To setup any other chained instance, include the chain flag in the command. To find out what's available, first issue a **ups list** command (see section 2.2 *Listing Product Information in a Database*). For example, before setting up the instance of **tex** declared as "old", make sure it's available, e.g.,:

```
% ups list -aOK+ tex
      "tex" "v3_1415a" "IRIX+5" "" "old"
      "tex" "v3_1415a" "OSF1+V3" "" "old"
```

If your machine's flavor corresponds to one that is listed, then you can go ahead and issue this **setup** command:

```
% setup -o tex
```

Unchained Instance

To setup an unchained instance of a product, include its version number. First, let's find one:

```
% ups list -aK+ -f IRIX+4 tex
      "tex" "v3_1415a" "IRIX+4" "" ""
```

If you're on an IRIX+4 machine, you can setup version v3_1415a of **tex** by issuing the command:

```
% setup tex v3_1415a
```

For more examples, see the reference section 22.1 *setup*.

2.4.2 When You Need to Specify Other Options

As simple as it is to use in everyday situations, **setup** is equipped with a host of options that allow you to specify precisely which instance to setup, and in some cases, how to set it up. We do not go into detail here about the more complex cases, but refer you to section 22.1 *setup*. However, it is worth mentioning two options: **-j** for setting up just the main product, and **-q** for specifying qualifiers.

The -j Option

By default, the **UPS** product dependencies are setup recursively along with the main product. Use the **-j** option to restrict the setup to *just* the specified product and none of its dependencies, e.g.,:

```
% setup -j exmh
```

The -q Option

To setup any product instance that has been declared with qualifiers, the exact set of qualifiers must be specified on the command line using the **-q** option. Qualifiers are case-sensitive, and must be entered on the command line exactly as they appear in the product declaration.

You can see whether the instance you want has one or more qualifiers by running **ups list**, for example:

```
% ups list -aK+ -f SunOS+5 gtools v2_1
"gtools" "v2_1" "SunOS+5" "" ""
"gtools" "v2_1" "SunOS+5" "build" ""
```

If you want to make sure you setup the second listed instance, specify **build** as a required qualifier:

```
% setup -q build gtools v2_1
```

If you're running this in a script and you would prefer to setup the instance with the qualifier, but it is not absolutely required, you can specify the qualifier as *optional* using a question mark and quotes, e.g.,

```
% setup -q "?build" gtools v2_1
```

In this case, you'll get the second listed instance above. But depending on the available instances in your case, the matched instance may or may not have the qualifier.

See the reference section 24.2.4 *-q* for more information on specifying qualifiers on the command line.

2.5 Running Unsetup on a Product

unsetup is intended to undo the changes to your software environment made during product setup. It makes the product no longer available for use. You may need to explicitly unsetup a **UPS** product if you are short on environment variable space and want to get rid of extra environment variables or shorten the `$PATH` variable length. Unsetup gets done automatically for you when you setup a different instance of the same product.

When you no longer need to access a product, in most cases you can simply type:

```
% unsetup <product>
```

for example:

```
% unsetup tex
```

If you just want to unsetup your main product and leave its product dependencies setup, include the **-j** option in your **unsetup** command, for example:

```
% unsetup -j <product>
```

If **unsetup** doesn't work as you expect for a particular product, see the reference section 22.2 *unsetup* for an explanation of how the behavior of **unsetup** depends on the product's implementation.

Part VI UPS and UPD Command Reference

Chapter 22: *UPS Command Reference*

This chapter contains full usage information on all the **UPS** commands. In particular, for each command you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a listing of commonly used options, without descriptions
- a listing of all valid options, with command-specific descriptions
- (as needed) a section called “Options Valid with -G”
- (as needed) a section called “More Detailed Description” which typically includes:
 - detailed command-specific usage information
 - information on environment variables that affect execution of the command or are affected by it
 - a list of internal functions the command performs (if more extensive than suggested by the command description)
- command examples

Chapter 23: *UPD/UPP Command Reference*

This chapter contains full usage information on all the **UPD** commands and the **UPP** command. In particular, for each command you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a listing of commonly used options, without descriptions
- a listing of all valid options, with command-specific descriptions
- (as needed) a section called “Options Valid with -G”
- (as needed) a section called “More Detailed Description” which typically includes detailed command-specific usage information
- command examples

For commands that have a corresponding **UPS** command, you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a reference to the corresponding **UPS** command
- a listing of any additional, **UPD**-specific options
- (as needed) command examples

For the `upd addproduct` and `upd install` commands we include a detailed list of the internal processes the command performs. The internal processes for the other commands can largely be inferred from these lists.

Chapter 24: *Generic Command Option Descriptions*

This chapter provides an alphabetical listing of **UPS/UPD** options with generic descriptions. More detailed information on a few selected options can be found at the end of the chapter.

In the command reference chapters, 22 and 23, the options supported by each command are listed with command-specific descriptions.

Chapter 25: *UPS/UPD Command Usage*

This chapter describes the syntax for **UPS** and **UPD** commands.

Chapter 26: *Product Instance Matching in UPS/UPD Commands*

When a **UPS** or **UPD** command is issued, the system must determine which product instance(s) to act upon. This determination is called *instance matching*. This chapter describes the algorithms used for instance matching.

Chapter 22: UPS Command Reference

This chapter contains full usage information on all the **UPS** commands. In particular, for each command you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a listing of commonly used options, without descriptions
- a listing of all valid options, with command-specific descriptions
- (as needed) a section called “Options Valid with -G”
- (as needed) a section called “More Detailed Description” which typically includes:
 - detailed command-specific usage information
 - information on environment variables that affect execution of the command or are affected by it
 - a list of internal functions the command performs (if more extensive than suggested by the command description)
- command examples

22.1 setup

Issue the **setup** command for a product prior to invoking the product. The **setup** command performs the necessary operations in your login environment to make an installed, declared product instance accessible to you. Typically, the operations include modifying environment variables or adding to your \$PATH.

22.1.1 Command Syntax

```
% setup [<options>] <product> [<version>]
```

22.1.2 Commonly Used Options

See section 22.1.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-z <databaseList>

22.1.3 All Valid Options

-? ("-" for csh) Prints command description and option usage information to screen

-B <depProdName>= "<options>"
Specifies options to prepend to the `setupRequired` line (in table file) for the dependent product **<depProdName>**

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-e Sets `$UPS_EXTENDED` (to the value **1**). This is useful for `SETUP` actions which call scripts.

-f <flavor> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under "The flavor options".

-j Ignores dependencies, sets up just specified top-level product

-k Prevents execution of `unsetup` files prior to (subsequent) `setup`

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

- n** Finds product instance chained to “new”
- o** Finds product instance chained to “old”
- O "<flags>"** Sets the value of \$UPS_OPTIONS to **<flags>**. This is useful for SETUP actions which call scripts.
- P** Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- R** Sets up only the required (non-optional) dependencies.
- s** Lists what command would do; but does not execute the command
- t** Specifies product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups` (relative to the product root directory)
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command (does not include time for sourcing of temp file for **setup**)

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

If a dependency is specified in the table file with a particular flavor, the flavor specified on the command line is ignored for that dependency.

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family.

If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.

- 0 Specifies flavor as NULL; equivalent to **-f NULL**
- 1 Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2 Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3 Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.1.4 More Detailed Description

In general, **UPS** products require that the **setup** command be issued on a product instance before invoking it (unless it is a dependent product of one that is already setup). The setup processes are intended to make the appropriate changes to your software environment in order to make the requested product available for use.

Only one instance of a product can be setup at a time. Each time you run **setup** on an additional instance of the same product, the previously active instance is automatically unsetup first.

Internal Processes

- Check node authorization
- If necessary, process UNSETUP action
- Process SETUP action
- Source the temp file

Environment Variables Set by Default During setup

When an instance is setup, either or both of the two environment variables `$<PRODUCT>_DIR` and `$SETUP_<PRODUCT>` may get defined. By default, both do.

<code>\$<PRODUCT>_DIR</code>	points to the root directory of the product instance selected by the setup command ¹
<code>\$SETUP_<PRODUCT></code>	a string containing all the information that the unsetup command needs in order to identify the instance when it is time to remove access to the product

1. In versions of **UPS** previous to v4, at **unsetup** time **UPS** matched the current definition of `$<PRODUCT>_DIR` with the product's versions in the database. If no match was found, it assumed `$<PRODUCT>_DIR/ups` as the location of the unsetup script.

In **UPS** v4, `$<PRODUCT>_DIR` may not be set because it is no longer a requirement. The unsetup action is generally not performed via a script, but rather via functions in a table file, and this table file is not constrained to reside under the product root directory.

In both of them, <PRODUCT> is the name of the product in upper case. We'll use the product **cern** as an example to show you typical values for these variables:

```
% setup cern
% echo $CERN_DIR
    /afs/fnal.gov/products/SunOS5/cern/v97aa
% echo $SETUP_CERN
    cern v97aa -f SunOS+5 -z /usr/upsII/ups_database/declared/afs
```

Use of the \$SETUP_<PRODUCT> Variable by unsetup

unsetup uses the environment variable \$SETUP_<PRODUCT>, by default, to determine which instance to unsetup. If this variable was *not* set during product setup (i.e., the **setup** default functions were not run, or **setupEnv()** was not run), then when you run **unsetup**, you must specify on the command line which instance to unsetup; running simply **unsetup <product>** causes no action to be taken. See *Use of the \$SETUP_<PRODUCT> Variable* under section 22.2 *unsetup* for more information regarding the **unsetup** command.

22.1.5 setup Examples



In the following examples, when we say that all dependencies of a product get set up, we mean all except optional dependencies that are unavailable.

Setup default instance of product

```
% setup xemacs
```

This sets up the current instance of the product **xemacs** for the best match flavor of your OS. If the product has any dependencies, they get setup too, by default.

```
% setup -v xemacs
```

This command sets up the same instance as above, but displays verbose information (usually used for debugging, but useful to see what's going on). If any file that is "opened for read" does not exist, you'll see **ERROR** at the end of the line. This is often but not always a fatal error. The output looks like this:

```
UPSFIL: /usr/upsII/ups_database/declared/oss/.upsfiles/dbconfig - Open file for read
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/current.chain - Open file for read
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/current.chain - Read 2 item(s)
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/v19_14.version - Open file for read
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/v19_14.version - Read 2 item(s)
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/v19_14.table - Open file for read
UPSFIL: /usr/upsII/ups_database/declared/oss/xemacs/v19_14.table - Read 4 item(s)
UPSFIL: /usr/upsII/ups_database/devel/new/xemacs/current.chain - Open file for read ERROR
```

Restrict the setup of dependent products

```
% setup -R exmh
```

Use of the **-R** option sets up the specified product and its required dependencies only.

```
% setup -j exmh
```

Use of the `-j` option sets up only the specified product; none of its dependencies get setup.

Setup a chained instance (other than the default “current”)

```
% setup -t tex
% setup -g test tex
```

Either of these commands sets up the instance of **tex** chained to “test” (for the default flavor). To setup any chained instance other than current, include the chain flag in the command.

Setup a product specifying its version

```
% setup tex v3_1415a
```

This command sets up version `v3_1415a` of **tex** whether or not it has a chain. Run a `ups list` command to get the version information.

Setup a product declared with qualifiers

```
% setup -q BUILD prod1
```

This command sets up the current instance of **prod1** for the operating system on which you’re working, along with its build dependencies (assuming the qualifier `BUILD` has been implemented in the product files in the standard way, see section 16.2.3 *Products Requiring Build (In-House and Third-Party)*).



Remember that qualifiers are case-sensitive.

Setup a product and activate extended functionality

To setup the instance of product **prod1** chained to development, and all of **prod1**’s dependencies, and to activate extended setup actions, enter:

```
% setup -d -e prod1
```

The `-e` option sets `$UPS_EXTENDED` on for **prod1** and for any of its **UPS** product requirements that were declared with the `-e` option. This is used to activate any extended functionality the product provider may have included in the setup action for this instance (e.g., defining extra environment variables).

Setup a product directly from CD-ROM

If you have a CD-ROM image that includes products that are unwound and ready-to-use, you can run them directly off the CD-ROM without downloading them to your system.

First insert the CD-ROM and run the `mount` command:

```
% mount -t iso9660 /dev/cdrom /path/to/db/on/cdrom
```

Then run the `setup` command. **UPS** needs to look in the right database, so either add it to your `$PRODUCTS` variable beforehand, or supply it on the command line, e.g.,:

```
% setup -z /path/to/db/on/cdrom <product> <version>
```

More information on **UPS** product distribution CD-ROMs can be found at <http://www.fnal.gov/docs/products/ups/ReferenceManual/misc/cdrom.html>.

22.2 unsetup

The **unsetup** command makes the specified product no longer available for use. It undoes the changes made to the environment by **setup**. You may need to explicitly unsetup a **UPS** product if you are short on environment variable space and want to get rid of extra environment variables or shorten the \$PATH variable length; otherwise you typically don't need to run this command.

22.2.1 Command Syntax

```
% unsetup [<options>] <product> [<version>]
```

22.2.2 All Valid Options

Typically, this command is issued with no options, and

- ? ("**-?**" for **cs**) Prints command description and option usage information to screen
- c Finds product instance chained to "current"
- d Finds product instance chained to "development"
- e Sets \$UPS_EXTENDED (to the value **1**).
- f <flavor> Described below under "The flavor options".
- g <chainName> Finds product instance chained to <chainName>
- H <flavor> Described below under "The flavor options".
- j Ignores dependencies, runs **unsetup** on just on top level product
- m <tableFileName> Specifies table file name
- M <tableFileDir> Specifies table file directory
- n Finds product instance chained to "new"
- o Finds product instance chained to "old"
- O "<flags>" Sets the value of \$UPS_OPTIONS to <flags>. (This option is ignored.)
- P Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList> Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir> Specifies the product root directory (This option is ignored.)
- s Lists what command would do; but does not execute the command
- t Finds product instance chained to "test"

- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which the product(s) are declared
- Z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

If a dependency is specified in the table file with a particular flavor, the flavor specified on the command line is ignored for that dependency.

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of `-0`, `-1`, `-2`, `-3`, **UPS** finds the product instance of specified level of that flavor; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.2.

22.2.3 More Detailed Description

unsetup is intended to undo the changes to your software environment made during product setup. It makes the product no longer available for use. You may need to explicitly **unsetup** a **UPS** product if you are short on environment variable space and want to get rid of extra environment variables or shorten the `$PATH` variable length. **Unsetup** gets done automatically for you when you setup a different instance of the same product.

When you no longer need to access a product, in most cases you can simply type:

```
% unsetup <product>
```

for example:

```
% unsetup tex
```

Sometimes this isn't sufficient. The `$SETUP_<PRODUCT>` variable governs the behavior, as described below.

Use of the `$SETUP_<PRODUCT>` Variable

unsetup uses the environment variable `$SETUP_<PRODUCT>`, by default, to determine which instance to **unsetup**. If this variable was *not* set during product setup (i.e., the **setup** default functions were not run, or **setupEnv()** was not run), then you must specify on the command line which instance to **unsetup**; running simply **unsetup** `<product>` causes no action to be taken.

If `$SETUP_<PRODUCT>` has been set (the usual case), it is best to run **unsetup** with no options (except possibly `-j` as discussed below). If any instance-identifying information besides product name is specified on the **unsetup** command line, this information gets ignored.

Behavior of **unsetup** for Product Dependencies

The behavior of **unsetup** as regards product dependencies depends upon a couple of factors:

- whether an **UNSETUP** action exists in the main product's table file
- whether `$SETUP_<PRODUCT>` has been defined for the product dependency

If `ACTION=UNSETUP` is defined for the main product, then¹:

- 1) if it includes the function **unsetupRequired** for the dependency with *no* instance-identifying information (e.g., **unsetupRequired** (`<dep_product>`) only; no options or version), and if `$SETUP_<PRODUCT>` is defined for the dependency, **unsetup** will be run on the instance identified by `$SETUP_<PRODUCT>`.
- 2) if it includes the function **unsetupRequired** for the dependency with *no* instance-identifying information (e.g., **unsetupRequired** (`<dep_product>`) only; no options or version), and if `$SETUP_<PRODUCT>` is *not* defined for the dependency, no **unsetup** will be performed for the dependency.

1. In the following numbered list, the same is true for the function **unsetupOptional**, with the difference that no failure will occur if the specified instance is not available to be **unsetup**.

- 3) if it includes the function **unsetupRequired** for the dependency with some instance-identifying information (e.g., **unsetupRequired -q "build" <dep_product>**), then **unsetup** is run on this specified instance of the product dependency; if **\$SETUP_<PRODUCT>** is defined for the dependency, it is ignored.

If **ACTION=UNSETUP** is *not* defined for the main product, then:

- 1) if **\$SETUP_<PRODUCT>** is defined for the product dependency, then **unsetup** will be run on the instance identified by **\$SETUP_<PRODUCT>**.
- 2) if **\$SETUP_<PRODUCT>** is *not* defined for the dependency, no **unsetup** will be performed for the dependency.

If you use the **-j** option in the **unsetup** command of the main product, only the main product gets unsetup; its product dependencies are left untouched.

Internal Processes

- Check node authorization
- Process UNSETUP action
- Source the temp file

The UNSETUP default functions are to undo the default SETUP functions (i.e., **unset \$<PRODUCT>_DIR** and **\$SETUP_<PRODUCT>**).

Note: If there is no UNSETUP action, then **unsetup** undoes everything done in SETUP action. However, if SETUP includes non-reversible functions, these cannot be undone by **unsetup**.

22.2.4 unsetup Examples

```
% unsetup tex
```

This command unsets the product **tex**. When you no longer need to access a product, in most cases you can simply use the product name to identify it.

```
% unsetup -j netscape
```

The **-j** option in this command causes **UPS** to unsetup the product **netscape**, while leaving all its dependencies setup.

22.3 ups configure

For any product instance whose table file includes a CONFIGURE action, the **ups configure** command executes this action. A CONFIGURE action usually includes functions to construct symbolic links, copy files, or perform automatic local customization of the product. The **ups configure** command gets run by default by **ups declare** when the product is initially declared to a database (see section 22.5 *ups declare*, in particular the **-C** option), but can be run manually as needed (e.g., on nodes of different flavors).

22.3.1 Command Syntax

```
% ups configure [<options>] <product> [<version>]
```

22.3.2 Commonly Used Options

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-z <databaseList>

22.3.3 All Valid Options

-? ("**-?**" for **csh**) Prints command description and option usage information to screen

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-f <flavor> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under "The flavor options".

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

-n Finds product instance chained to "new"

-o Finds product instance chained to "old"

-O "<flags>" Sets the value of **\$UPS_OPTIONS** to **<flags>**.

-P Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)

- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of `-0`, `-1`, `-2`, `-3`, **UPS** finds the product instance of specified level of that flavor; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.2.

22.3.4 More Detailed Description

Installation/configuration procedures that can be completely automated are typically collected in the table file in a CONFIGURE action (*actions* are described in Chapter 33: *Actions and ACTION Keyword Values*), or in a script called `configure` which is called from the table file. The configuration may involve creating links to the product root directory from other areas. If the area is not identical for each machine flavor accessing the UPS database in which the product instance has been declared (i.e., the same path but separate areas), then you will need to run the **ups configure** command manually once per flavor, on a node of that flavor. If each node mounts a unique area, you generally have to run special commands (e.g., **ups install**, **ups initialize**, etc.) that are documented in the product's `INSTALL_NOTE` file. If you are not sure whether you need to configure a product instance on each flavor/node, look through the configuration steps in the table file to see what they do.

Internal Processes

- Check node authorization
- Process CONFIGURE action
- Execute temp file

22.3.5 ups configure Examples

perl is a product that requires **ups configure** to be run manually for each machine flavor in a cluster. The sample command, which should be issued from a machine of flavor SunOS+5, runs the CONFIGURE action in the table file associated with the product **perl**, version v5_005 for flavor SunOS+5.

```
% ups configure perl v5_005 -f SunOS+5
```

This command should take care of the configuration of **perl** on all the machines of flavor SunOS+5 in the cluster. A command like this, but with the appropriate flavor, must be run for each machine flavor represented in the cluster.

22.4 ups copy

The `ups copy` command was designed as a **UPS** product development tool allowing a new instance of a product to be declared “like” another.

22.4.1 Command Syntax

```
% ups copy -G "<ups_declare_options> <product> <version>" \  
  [<options>] <product> <version>
```

22.4.2 Commonly Used Options

See section 22.4.3 *All Valid Options* for descriptions of each option.

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (alone or together with one of  
                      -0, -1, -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-G "<options>"  
-q <qualifierList>  
-r <prodRootDir>  
-T <path or URL>  
-W  
-X  
-z <databaseList>
```

22.4.3 All Valid Options

```
-? ("-?" for csh)    Prints command description and option usage information to screen  
-c                  Finds source product instance chained to “current”  
-d                  Finds source product instance chained to “development”  
-f <flavor>         Described below under “The flavor options”.  
-g <chainName>      Finds product instance chained to <chainName>  
-G "<options>"       Specifies options to be passed to the ups declare command for  
                      target product instance; see below  
-H <flavor>         Described below under “The flavor options”.  
-m <tableFileName> Specifies table file name of source product instance  
-M <tableFileDir>  Specifies table file directory of source product instance  
-n                  Finds source product instance chained to “new”
```

- o** Finds source product instance chained to “old”
- O "<flags>"** Sets the value of \$UPS_OPTIONS to <flags>.
- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory of source product instance
- t** Finds source product instance chained to “test”
- T <path or URL>** Specifies archive file path or URL. This is used only for declarations in distribution databases for which products are maintained in tar or gzip (archived) format.
- U <upsDir>** Specifies location of `ups` directory of source product instance; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- W** Uses environment variables (e.g., \$SETUP_<PRODUCT>) to identify dependent product instances for target product (that is, it uses instances that are already setup in preference to what is listed in table file)
- X** Executes the `ups declare` command instead of just echoing it
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavor>** Finds source product instance of specified flavor, and sets target instance’s flavor on distribution node, unless overridden in `-G`. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, the best match is picked for source instance. This also determines target instance flavor unless overridden in `-G`.

If used with any of `-0`, `-1`, `-2`, `-3`, instance with specified level of that flavor is picked as source instance; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`

- 1 Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2 Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3 Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.4.4 Options Valid with -G

In order to distinguish the target product instance from the source, the declarations for the two instances must differ by at least one instance-identifying element. The **-G** option provides the means to specify the target instance identifiers; it takes a list of **ups declare** command line elements as an argument. Any identifier not specified via **-G** retains the value of the source instance. The elements valid for use with **-G** include **<product>**, **<version>** and the following subset of the **ups declare** options:

-A <nodeList>, **-c**, **-d**, **-D <origin>**, **-f <flavor>**, **-g <chainName>**, **-n**, **-o**, **-O "<flagList>"**, **-p "<description>"**, **-q <qualifierList>**, **-t**, **-z <databaseList>**, **-0**, **-1**, **-2**, **-3**

See section 22.5 *ups declare* for details on each option. If the argument to **-G** includes the product version, the product name must be included ahead of the version; the first unflagged element is always interpreted as the product name and the second as the version.

22.4.5 More Detailed Description

The command **ups copy** is intended mainly for product developers declaring new instances on their development systems. It simplifies declaration of new instances of products that already exist in a **UPS** database. There is no restriction against using **ups copy** to copy the installation of a different product, however it's usually not particularly helpful in that situation.

Notes:

- **ups copy** runs **ups declare** if you use the **-X** (uppercase **-x**) option; if not used, the declare command is just echoed.
- Use the **-G** option to specify declaration information that is to be different from the installation you're using as a model. At least one instance-identifying element must be specified using **-G** to distinguish the source from the target instance.
- If you use the option **-w**, you will pick up the current environment. For example, if the previously declared instance depends on **v1_0** of some product (e.g., **joe v1_0**), but the new instance should have **joe v2_0** as a dependency, first run **setup joe v2_0**, then run **ups copy** with **-w**.

Internal Processes

- Process COPY action
- Create a table file entry for new instance (may use environment for **UPS** product requirements)
- If simulation only, write table file entry to temp and echo appropriate declare command
- Otherwise, write/merge in table file and declare new instance (see **ups declare** internals)
- Execute temp file

22.4.6 ups copy Examples

```
% ups copy dog v1 -G "dog v3 -f SunOS -q test -m v3.table -M ups \  
-r /path/to/dog/v3"
```

This command runs a **ups copy** command for **dog** version v3, without the **-X** option so that the **ups declare** command just gets echoed, not executed. The **-G** argument here gives the product name and version plus options to be used by the **ups declare** command: the flavor (**-f SunOS**), a qualifier (**-q test**), the table file name and location (given via **-m** and **-M**), and the product root directory (given via **-r**). The command output looks like this:

```
/var/tmp/baaa006ni_table_dog  
ups declare dog v3 -f "SunOS" -q "test" -r "/path/to/dog/v3" \  
-U "ups" -m "v3.table" -M "ups"
```

The second command is identical to the first example except that the **-X** option instructs it to execute the **ups declare** command:

```
% ups copy dog v1 -G "dog v3 -f SunOS -q test -r /path/to/dog/v3 \  
-M ups -m v3.table" -X
```

22.5 ups declare

The **ups declare** command is used for two separate purposes:

- 1) to initially declare an instance to a database (and optionally add a chain at the same time)
- 2) to add a chain to a previously declared instance

22.5.1 Command Syntax

For initially declaring an instance

```
% ups declare <flavor_option> -r <prodRootDir> [<other_options>] \  
  <product> <version>
```

For declaring a chain

```
% ups declare <chain_option> [<other_options>] <product> \  
  <version>
```

22.5.2 Commonly Used Options

See section 22.5.3 *All Valid Options* for descriptions of each option.

For initially declaring an instance

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                    -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-m <tableFileName>  
-q <qualifierList>  
-r <prodRootDir>  
-z <databaseList>
```

For declaring a chain

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                    -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-q <qualifierList>  
-z <databaseList>
```

22.5.3 All Valid Options

Valid only for initially declaring an instance (not for assigning a chain)

- A <nodeList>** Specifies nodes authorized to access the product; sets the keyword AUTHORIZED_NODES
- b <compileFile>** Specifies name of the output file for the **ups compile** command (described in Chapter 37: *Use of Compile Scripts in Table Files*); sets the keyword COMPILE_FILE
- D "<origin>"** Specifies the product's master source file; sets the keyword ORIGIN (all spaces get removed from **<origin>** for the keyword value)
- L** Adds the STATISTICS keyword to the version file, thereby instructing **UPS** to keep statistics on this product instance. A record of the form:

```
"tcl" "v7_3q" "IRIX" "" "" "berman" "1998-03-13 17.56.54 GMT" "list"
```

will get added to the file
\$PRODUCTS/.upsfiles/statistics/<product> each time a **UPS** command is run on this instance.
- u <compileDir>** Specifies the directory for the output file (which is named via the **-b** option) for the **ups compile** command; sets the keyword COMPILE_DIR.

Valid for both functions

- ? (" -?" for csh)** Prints command description and option usage information to screen.
- c** Chains the product instance to "current"; when this chain gets declared, the corresponding ACTION=CURRENT in the table file gets executed, if it exists.
- C** When initially declaring a product, **-C** prevents execution of the CONFIGURE action.
When declaring a chain, **-C** prevents execution of the corresponding chain action.
- d** Chains the product instance to "development"; when this chain gets declared, the corresponding ACTION=DEVELOPMENT in the table file gets executed, if it exists.
- f <flavor>** Described below under "The flavor options".
- g <chainName>** Chains the product instance to **<chainName>** (this is useful for user-defined chains). When any chain gets declared, the corresponding ACTION=<CHAIN_NAME> in the table file gets executed.
- H <flavor>** Described below under "The flavor options".
- m <tableFileName>** Specifies table file name; when initially declaring a product, sets the keyword TABLE_FILE.

- M <tableFileDir>** Specifies table file directory; when initially declaring a product, sets the keyword TABLE_DIR. Specify only if file is not in one of the two default locations, namely under \$PRODUCTS/<product> or in the ups directory.
- n** Chains the product instance to “new”; when this chain gets declared, the corresponding ACTION=NEW in the table file gets executed, if it exists.
- o** Chains the product instance to “old”; when this chain gets declared, the corresponding ACTION=OLD in the table file gets executed, if it exists.
- O "<flags>"** Sets the value of \$UPS_OPTIONS to <flags>.
- q <qualifiers>** When initially declaring a product, **-q** specifies required and/or optional qualifiers to include in the declaration, and sets the keyword QUALIFIERS.
When adding a chain, **-q** specifies required and/or optional qualifiers to identify the instance.
- r <prodRootDir>** Specifies the product root directory; when initially declaring a product, sets the keyword PROD_DIR.
A note for developers: you may find it convenient to use the construction **-r \pwd** if you’re working in the product root directory.
- t** Chains the product instance to “test”; when this chain gets declared, the corresponding ACTION=TEST in the table file gets executed, if it exists.
- T <path or URL>** Specifies archive file path or URL. This is used only for declarations in distribution databases for which products are maintained in tar or gzip (archived) format. When initially declaring a product, it sets the keyword ARCHIVE_FILE.
- U <upsDir>** Specifies location of ups directory; default value is ups; when initially declaring a product, sets the keyword UPS_DIR
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database in which to declare the product (see section 26.1 *Database Selection Algorithm*); or, if adding a chain, specifies the database(s) in which product is declared
- Z** Times the command

The flavor options

Flavor may be specified using **-f**, or using **-H** in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Declares product instance as specified flavor; when initially declaring a product, sets the keyword FLAVOR.

- H <flavor>** Must be used with any of **-0**, **-1**, **-2**, **-3**. Specifies flavor and builds a flavor list for that family starting at the level specified. **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.5.4 More Detailed Description

Declaring an Instance for the First Time

In the **ups declare** command:

- **-C** prevents execution of the CONFIGURE action, if any, and is normally not included. The default (and usually desired) behavior is to execute the CONFIGURE action.
- You must include a flavor specification, there is no default flavor when a product is first declared. It sets the value of the FLAVOR keyword.
- Most products require a table file. The table file must exist before running **ups declare!** In most cases you need to include the table file name (**-m**). (Since in a very few cases a table file isn't required, **-m** is not a strictly required option.)
- If the product's table file was placed in either of the two default locations (under `/path/to/database/<product>` or in the product's `ups` directory), then **-M <table_file_dir>** is not needed. Only use the **-M** option if you have moved the table file to a separate location where **UPS** won't otherwise find it.
- In most cases you need to include product root directory (**-r**). Exceptions include wrapper products which consist only of a table file, and thus have no product root directory.
- If the product's `ups` directory tar file was unwound in the default location (`$(PRODUCT)_DIR/ups`), then **-U <ups_dir>** is not needed. If the `ups` directory is located elsewhere (or named differently), this specification must be included. In general, you should not include this qualifier.

- If you choose not to specify the target database explicitly (**-z <database>**), **UPS** chooses it automatically via the **\$PRODUCTS** variable. If **\$PRODUCTS** points to multiple databases, you need to be a little careful about database selection. The database matching algorithm is described in section 26.1 *Database Selection Algorithm*.

If the product has dependencies that are declared in different databases, **UPS** must be able to find all of them in order to resolve the dependencies. You can rely on **\$PRODUCTS** if all the necessary databases are included in it. Otherwise specify them on the command line (e.g., **-z <database1>:<database2>:...>** or **-z \$PRODUCTS:<database1>:<database2>:...>**).

- You may include chain information on this command. See the description below.

Adding Chains to an Existing Instance

When you add one or more chains to an existing instance, **UPS** doesn't allow you to change anything else about that instance. Regarding the options to specify:

- You of course need to specify the chain or chains to add using either **-g <chain_name>** or one of **-c**, **-d**, **-n**, **-o**, **-t**.
- **-C** prevents the corresponding chain action(s) in the table file from being executed.
- You do not strictly need to specify flavor. **UPS** will default to the best flavor match (described in section 26.2.4 *Flavor and Qualifier Matching Algorithm*). You can override the default using **-f** or one of the number options (**-0**, **-1**, **-2**, **-3**).
- Specify qualifiers (**-q**) as necessary to select the appropriate instance.
- Specify the database (**-z**) as necessary to select the appropriate instance.

Internal Processes

- Find database to use
- If necessary, process 'UNCHAIN' action
- Process DECLARE action
- If necessary, process CONFIGURE action
- If necessary, warn if there is a TAILOR action
- If necessary, process the 'CHAIN' action
- If necessary, warn if there are START/STOP actions
- If current chain, try to copyman, copycatman and copyInfo files
- Execute the temp file
- If successful, modify all appropriate files on disk

22.5.5 ups declare Examples

Declare a product with no chain using defaults where possible

```
% ups declare myprod v1_0 -f Linux+2 -m myprod.table \  
-r /path/to/myprod/v1_0/Linux+2
```

UPS finds the product in the directory given by the `-r` option, and declares it to a database in `$PRODUCTS` according to the selection algorithm discussed in 26.1 *Database Selection Algorithm*. The product instance gets declared with the specified name, version and flavor, and no qualifiers. UPS looks for the table file, called `myprod.table` in the two default locations (command fails if table file doesn't exist, or is not found in either location).

Declare a product with no chain using defaults where possible

```
% ups declare myprod2 v1_0 -2 -g test -m myprod2.table \  
-r myprod2/v1_0/IRIX+6 -z /my/local/db:$PRODUCTS
```

For this example, we assume the local machine flavor is IRIX+6.2. UPS finds the product in the directory given by the `-r` option. The specified path is taken relative to `PROD_DIR_PREFIX`. UPS declares the product to one of the listed databases according to the selection algorithm discussed in 26.1 *Database Selection Algorithm*. Each of the dependencies, if any, must exist in at least one of the listed databases.

The product instance gets declared with the specified name, version, no qualifiers, and the chain "test". The flavor declaration is the level `-2` specification of the machine, namely IRIX+6. UPS looks for the table file, called `myprod2.table` in the two default locations (command fails if table file doesn't exist, or is not found in either location).

Add a chain to a previously declared instance

```
% ups declare myprod2 v1_0 -2 -g test -m myprod2.table \  
-r myprod2/v1_0/IRIX+6 -z /my/local/db:$PRODUCTS  
  
% ups declare myprod2 v1_0 -2 -c -z /my/local/db:$PRODUCTS
```

This command declares the product instance of the previous example "current" (via the `-c` option). Generally, a product is first declared as "test", and then after a "debugging period" (often several weeks), an updated release is cut and chained to "current". Notes:

- In most UPS/UPD commands you specify either chain (often defaulted to "current") or version. Here you need both: the version is required to identify the instance, and the chain is required because it is being assigned.
- You should specify the database `/my/local/db` first since that's how it was initially declared. UPS will traverse the databases in the same order to find the right instance.

22.6 ups depend

The **ups depend** command lists product dependencies of the specified product instance(s) as declared in the (local) database. On user nodes it is generally used to determine what products will get setup along with the “parent” product. **UPD** uses it on product servers to determine what dependencies to install.

22.6.1 Command Syntax

```
% ups depend [<options>] <product> [<version>]
```

22.6.2 Commonly Used Options

See section 22.6.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-j

-K <keywordList>

-l

-q <qualifierList>

-R

-z <databaseList>

22.6.3 All Valid Options

-? (" -?" for csh) Prints command description and option usage information to screen

-c Finds product instance chained to “current”

-d Finds product instance chained to “development”

-f <flavor> Described below under “The flavor options”.

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under “The flavor options”.

-j Ignores lower level dependencies, finds direct dependencies of top level product only

-K <keywordList> Returns values of specified keywords only; valid keywords are listed in section 27.4 *List of Supported Keywords*

-l Produces a long listing including all the table file functions that would be executed in a **setup** command.

- m** <tableFileName> Specifies table file name
- M** <tableFileDir> Specifies table file directory
- n** Finds product instance chained to “new”
- o** Finds product instance chained to “old”
- q** <qualifierList> Finds product instance with the specified qualifiers (required and/or optional)
- r** <prodRootDir> Specifies the product root directory
- R** Lists only the required (non-optional) dependencies.
- t** Finds product instance chained to “test”
- U** <upsDir> Specifies location of `ups` directory; default value is `ups`
- v**(vvv) Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z** <databases> Specifies the database(s) to search
- Z** Times the command

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f** <flavor> Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H** <flavor> Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.6.4 ups depend Examples

Execute command using all default values (no options)

```
% ups depend exmh
```

The first example requests information for the default instance of **exmh**, which is the one chained to “current” for the best-match flavor of the machine (SunOS+5 for this example). The output looks like this:

```
exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__tk v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | |__tcl v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__mailtools v2_3 -f NULL -z /afs/fnal.gov/ups/db -g current
|__mimetools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__glimpse v3_0a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__www v3_0 -f NULL -z /afs/fnal.gov/ups/db -g current
| |__netscape v4_05 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| | |__ghostview v4_0 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| | |__ximagetools v4_0 -f NULL -z /afs/fnal.gov/ups/db -g current
| | |__imagemagick v4_04 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | |__xfig v3_20 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | |__xanim v2_70_64 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | |__xpdf v0_7 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__lynx v2_8_1 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__ispell v3_1b -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
```

If a chain rather than a version number is used to specify the instance (as is the case for the default “current” instance), then the chain appears in the output line for the product (notice the **-g current** in the first line); otherwise the chain is not listed (compare to the following example). Compare this example to the following one:

```
% ups depend exmh v2_0_2
```

The instance is the same, the difference is that it was specified using the version rather than the chain (output edited for brevity):

```
exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__tk v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | |__tcl v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
...
```

List only required dependencies

```
% ups depend -R exmh
```

We use the **-R** option to request only the dependencies listed as “required” for the same product instance as in the previous examples:

```

exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__tk v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
| |__tcl v8_0_2 -f SunOS+5 -z /afs/fnal.gov/ups/db
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
| |__mailtools v2_3 -f NULL -z /afs/fnal.gov/ups/db -g current
|__mimertools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current

```

List a subset of fields for required dependencies only

```
% ups depend -RK product:version:flavor exmh
```

To return a subset of the output fields, we include the **-K** option (described in section 24.2.3 *-K*):

```

"exmh" "v2_0_2" "NULL"
"expect" "v5_25" "SunOS+5"
"tk" "v8_0_2" "SunOS+5"
"tcl" "v8_0_2" "SunOS+5"
"mh" "v6_8_3c" "SunOS+5"
"mailtools" "v2_3" "NULL"
"mimertools" "v2_7a" "SunOS+5"

```

List only direct dependencies

```
% ups depend -j exmh
```

We use the **-j** option to request “just” the direct dependencies of the specified product (dependencies of dependencies are omitted):

```

exmh v2_0_2 -f NULL -z /afs/fnal.gov/ups/db -g current
|__expect v5_25 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__mh v6_8_3c -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__mimertools v2_7a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__glimpse v3_0a -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__www v3_0 -f NULL -z /afs/fnal.gov/ups/db -g current
|__netscape v4_05 -f SunOS+5 -z /afs/fnal.gov/ups/db -g current
|__ispell v3_1b -f SunOS+5 -z /afs/fnal.gov/ups/db -g current

```

22.7 ups exist

The `ups exist` command is used to test whether a `setup` command issued with the same command line elements is likely to succeed. It was designed primarily for use in scripts.

22.7.1 Command Syntax

```
% ups exist [<options>] <product> [<version>]
```

22.7.2 Commonly Used Options

See section 22.7.3 *All Valid Options* for descriptions of each option.

`-f <flavor>` Or one of `-0`, `-1`, `-2`, `-3`, or `-H` (alone or together with one of `-0`, `-1`, `-2`, `-3`)

`-g <chainName>` Or one of `-c`, `-d`, `-n`, `-o`, `-t`

`-j`

`-q <qualifierList>`

`-z <databaseList>`

22.7.3 All Valid Options

`-? (" -?" for csh)` Prints command description and option usage information to screen

`-B <depProdName>= "<options>"`
Specifies options to prepend to the `setupRequired` line (in table file) for the dependent product `<depProdName>`

`-c` Finds product instance chained to “current”

`-d` Finds product instance chained to “development”

`-e` Sets `$UPS_EXTENDED` (to the value `1`).

`-f <flavor>` Described below under “The flavor options”.

`-g <chainName>` Finds product instance chained to `<chainName>`

`-H <flavor>` Described below under “The flavor options”.

`-j` Ignores dependencies, operates just on top level product

`-k` Prevents execution of unsetup files prior to (subsequent) setup

`-m <tableFileName>` Specifies table file name

`-M <tableFileDir>` Specifies table file directory

`-n` Finds product instance chained to “new”

- o** Finds product instance chained to “old”
- O "<flags>"** Sets the value of \$UPS_OPTIONS to **<flags>**.
- P** Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

-3

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.7.4 More Detailed Description

The **ups exist** command is used to test whether a **setup** command issued with the same command line elements is likely to succeed. As for all the **UPS/UPD** commands, if the **setup** command finds a corresponding action in the selected table file, it

- 1) translates the functions listed under the action into shell commands,
- 2) writes them to a temporary script in \$TMPDIR (if \$TMPDIR isn't set, the default is /tmp), and
- 3) invokes the script to execute the shell commands.

ups exist checks for a properly declared matching instance, and verifies that you have the necessary permissions to create the temporary script. If so, it creates the script, but it does not execute it.

In the C shell family **ups exist** sets the \$status variable to 0 if it was able to create the temporary file, or to 1 for error. In the Bourne shell family, it sets the \$? variable similarly.

This command is rarely used from the command line, and is more useful in scripts where a failed setup could cause the script to abort. When issued from the command line, it returns no output if the command succeeds.

Internal Processes

- Check node authorization
- If necessary, process UNSETUP action
- Simulate SETUP action

22.7.5 ups exist Examples

This command is rarely used from the command line, and is more useful in scripts where a failed setup could cause the script to abort. When issued from the command line, it returns no output if the command succeeds.

In the C shell family **ups exist** sets the \$status variable to 0 if it was able to create the temporary file, or to 1 for error. In the Bourne shell family, it sets the \$? variable similarly. As an example, we can run **ups list** and find that there is a current instance of the product **tex** for the flavor IRIX+6 but not for IRIX+6.2. Running **ups exist** for each flavor, we see that the variables get set accordingly. For the C shell family:

```
% ups exist tex -f IRIX+6; echo $status
```

```
0
```

```
% ups exist tex -f IRIX+6.2; echo $status
```

```
1
```

For the Bourne shell family:

```
$ ups exist tex -f IRIX+6; echo $?
```

0

```
$ ups exist tex -f IRIX+6.2; echo $?
```

1

22.8 ups flavor

The **ups flavor** command with no options returns the flavor of the machine. If a flavor level is specified (e.g., **-0**, **-1** ...), it returns the flavor according to that level. **ups flavor** generates a flavor table if the **-H** option is used. The flavor levels and the term *flavor table* are defined in section 22.8.4 *More Detailed Description*.

22.8.1 Command Syntax

```
% ups flavor [<options>]
```

22.8.2 Commonly Used Options

See section 22.8.3 *All Valid Options* for descriptions of each option.

-f <flavor>	Or one of -0 , -1 , -2 , -3
-H <flavor>	Alone or together with one of -0 , -1 , -2 , -3
-1	

22.8.3 All Valid Options

-? (" -? " for cs h)	Prints command description and option usage information to screen.
-f <flavor>	Specifies flavor.
-H <flavor>	Specifies host flavor family from which to build a flavor table. If used with any of -0 , -1 , -2 , -3 , specifies corresponding level of specified host flavor family.
-1	Produces a flavor table.
-v(vvv)	Prints out extra debugging information.
-z	Times the command.
-0	Specifies flavor as NULL; equivalent to -f NULL
-1	Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to -f SunOS ; if given together with -H IRIX+6.2 , flavor is then specified as IRIX.
-2	Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to -f SunOS+5 ; if given together with -H IRIX+6.2 , flavor is then specified as IRIX+6.

-3 Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.8.4 More Detailed Description

The **ups flavor** command returns flavor information about the machine issuing the command, or for a flavor requested via the **-H** option. When entered with no options, the command returns the full OS specification of the machine.

When entered with the **-l** (long) option, **ups flavor** returns what we call a *flavor table*, which is a list including every level of specificity for a flavor that you could use to find or declare a product instance. For example, on a SunOS machine (release 5.6) it outputs:

```
% ups flavor -l
```

```
SunOS+5.6  
SunOS+5  
SunOS  
NULL  
ANY
```

If **-H <flavor>** is used with **-l**, **ups flavor** builds a flavor table for the flavor given by **-H**. This is useful if you're not sure what levels are allowable for a particular basic flavor. The flavor table lists flavors starting at the level you specify. Compare the following two commands and output:

```
% ups flavor -lH IRIX+6.6
```

```
IRIX+6.6  
IRIX+6  
IRIX  
NULL  
ANY
```

```
% ups flavor -lH IRIX
```

```
IRIX  
NULL  
ANY
```

You can specify a particular level using the number options: **-0**, **-1**, **-2**, **-3**, of which **-3** is the most highly specified; for example:

```
% ups flavor -3
```

```
SunOS+5.6
```

```
% ups flavor -2
```

```
SunOS+5
```

```
% ups flavor -1
```

```
SunOS
```

```
% ups flavor -0
```

```
NULL
```

22.8.5 ups flavor Examples

Find full flavor specification of machine

```
% ups flavor
```

This command returns the full OS specification of the machine, for example:

```
SunOS+5.6
```

Create a flavor table for machine's OS

```
% ups flavor -1
```

This command returns a flavor table for the flavor of the machine. For example, on a SunOS+5.6 machine it outputs:

```
SunOS+5.6  
SunOS+5  
SunOS  
NULL  
ANY
```

Find flavor specification of machine, at different levels

```
% ups flavor -3
```

The `-3` option requests the machine's flavor as the most significant OS specification or the full specification, e.g.,:

```
SunOS+5.6
```

```
% ups flavor -1
```

The `-1` option requests the machine's flavor as the OS value up to the generic OS, e.g.,:

```
SunOS
```

```
% ups flavor -0
```

This always returns the NULL string.

Create a flavor table for host flavor, at different levels

```
% ups flavor -1H IRIX+6.6
```

This creates a flavor table listing flavors starting at the level specified via `-H`, in this case "level 3":

```
IRIX+6.6  
IRIX+6  
IRIX  
NULL  
ANY
```

```
% ups flavor -1H IRIX+6
```

This creates a flavor table listing flavors starting at the level specified via `-H`, in this case "level 2":

```
IRIX+6  
IRIX  
NULL  
ANY
```


22.9 ups get

The `ups get` command is rarely used by anyone except product developers/maintainers. Currently it can only be used with the `-F` option. `ups get -F` lists any files on the local node which were distributed with the specified product instance(s) and which are maintained outside of the product root directory. The list does not include table files, for which the location is maintained in the version file.

The `ups get` command was designed primarily for use by `UPD`, which calls it internally. As such it is rarely used outside of that context. In a future release, `ups get` may acquire additional functions.

22.9.1 Command Syntax

```
% ups get -F [<other_options>] <product> [<version>]
```

22.9.2 All valid options

- `-?` ("`-?`" for `cs`) Prints command description and option usage information to screen
- `-c` Finds product instance chained to "current"
- `-d` Finds product instance chained to "development"
- `-f <flavor>` Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but `UPS` looks only at first in list.
- `-F` Prints to screen a list of files that are associated with the product but which are maintained external to the products area (excluding table file)
- `-g <chainName>` Finds product instance chained to `<chainName>`
- `-H <flavor>` Specifies flavor and builds a flavor list for that family starting at the level specified. `UPS` finds the best match instance for the specified flavor family. Multiple values accepted, but `UPS` looks only at first in list.
- `-m <tableFileName>` Specifies table file name
- `-M <tableFileDir>` Specifies table file directory
- `-n` Finds product instance chained to "new"
- `-o` Finds product instance chained to "old"
- `-q <qualifierList>` Finds product instance with the specified qualifiers (required and/or optional)
- `-r <prodRootDir>` Specifies the product root directory
- `-t` Finds product instance chained to "test"

- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command

22.9.3 ups get Example

```
% ups get -F ups
```

In the database on the machine used, the **UPS** product has a few files maintained outside of the product root directory. This command returns the output:

```
/fnal/ups/db/.upsfiles/configure/v4_4a_OSF1+V4_/current  
/fnal/ups/db/.upsfiles/configure/v4_4a_OSF1+V4_/uncurrent  
/fnal/ups/db/.upsfiles/configure/v4_4a_OSF1+V4_/unconfigure
```

22.10 ups help

The `ups help` command lists all the **UPS** commands with brief definitions. There are no options for this command.

22.10.1 ups help Example

```
% ups help
```

```
UPS commands:
```

```
for specific command options use "ups COMMAND -?"
```

```
configure : Environmentally configure a product instance.
copy       : Allow one instance of a product to be declared "like" another.
declare    : Add a product instance or a chain to a UPS Database.
depend     : List (for a specified UPS product instance) UPS product
            requirements or all UPS product instances that have the
            specified UPS product instance as a requirement.
exist      : Determine if a setup command with the same options
            would likely succeed.
flavor     : Print flavor of a machine, optionally by level, or table
            generated for searching
get        : Return a list of all files that are needed by a product
            instance and do not live under the product root directory.
help       : Output help information for all UPS commands
list       : List UPS Database information about product instances.
modify     : Allow editor modification of the UPS Database files.
            The altered files are verified before being rewritten.
setup      : Prepare the environment in order to be able to use a product
            instance.
start      : Perform any necessary actions for a product instance at system
            boot.
stop       : Perform any necessary actions for a product instance at
            system shutdown.
tailor     : Perform any product instance tailoring that needs to be done
            once (specify hardware device location) or needs user input.
touch      : Will change a ups file modify time (MODIFIED) to current time
            (it will probably also change the modifier (MODIFIER)).
unconfigure : Undo any actions performed by the configure command.
undeclare  : Remove a product instance from a UPS Database.
            if chain(s) are specified ONLY the chain(s) will version will
            be removed
unsetup    : Return the environment to a pre-setup state.
verify     : Check the specified instances for correct formatting and
            information.
```


22.11 ups list

The `ups list` command returns information about the declared product instances in a **UPS** database. Two output styles are provided: a formatted one that is easy for users to read, and a condensed one for parsing by a subsequent command or a script.

22.11.1 Command Syntax

```
% ups list [<options>] [<product>] [<version>]
```

22.11.2 Commonly Used Options

See section 22.11.3 *All Valid Options* for descriptions of each option.

- `-a`
- `-f <flavorList>` Or one of `-0`, `-1`, `-2`, `-3`, or `-H` (alone or together with one of `-0`, `-1`, `-2`, `-3`)
- `-g <chainName>` Or one of `-c`, `-d`, `-n`, `-o`, `-t`
- `-K <keywordList>`
- `-l`
- `-q <qualifierList>`
- `-z <databaseList>`

22.11.3 All Valid Options

- `-? ("-" for csh)` Prints command description and option usage information to screen
- `-a` Lists all instances that match the other options given on command line
- `-c` Finds product instance(s) chained to “current”
- `-d` Finds product instance(s) chained to “development”
- `-f <flavorList>` Described below under “The flavor options”.
- `-g <chainName>` Finds product instance(s) chained to `<chainName>`
- `-H <flavorList>` Described below under “The flavor options”.
- `-K <keywordList>` Returns values of specified keywords only; valid keywords are listed in section 27.4 *List of Supported Keywords*
- `-l` Produces a long listing; lists all keywords for the product instance(s)
- `-m <tableFileName>` Specifies table file name
- `-M <tableFileDir>` Specifies table file directory
- `-n` Finds product instance(s) chained to “new”

- o** Finds product instance(s) chained to “old”
- q <qualifierList>** Finds product instance(s) with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- t** Finds product instance(s) chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product instance(s)
- z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavorList>** Finds product instance(s) of specified flavor. If specified and no exact match is found, the command fails. If multiple values specified, **UPS** looks for instances matching all the values.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified.

Can be used alone (without an accompanying number option). **UPS** looks for instances matching all the flavor levels. If multiple values specified (usually of different flavor families), **UPS** looks for instances matching all the flavor levels of each value.

If used with any of `-0`, `-1`, `-2`, `-3`, **UPS** looks for product instances of specified level of that flavor; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`. If multiple values specified (usually of different flavor families), **UPS** looks for instances matching each value, according to the accompanying number option.

Note: if `-a` and `-H` are both used, and `-H` is used without a number option, then `-H` has no effect; all flavors get listed.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.11.4 More Detailed Description

ups list is useful for finding out what products are in the database that you use, what the current version of a product is for your machine's flavor, and other information. Product installers and other administrative users can use it to get detailed information about a product's installation and to find product files.

You can specify the information you want contained in the output by including various options in the command. As is standard in **UPS**, if no chain, version or flavor is specified, and **-a** (for all instances) is *not* specified, **UPS** returns only the instance declared as current for the best-matched flavor of the requesting machine.

Formatted Output Style

One output style is for visual parsing (this is the default output, which we call *formatted*), with output that looks like:

```
% ups list xemacs
      DATABASE=/usr/upsII/ups_database/declared/oss
      Product=xemacs Version=v19_14 Flavor=SunOS+5
      Qualifiers="" Chain=current
```

Notice that the product name, version, flavor, qualifiers and chain(s) are the default fields that get returned. The database (first line of the output) is included as a header, not as part of the per-instance data. Each piece of data returned for an instance is preceded by its keyword for identification.

You cannot choose arbitrary output fields for the selected instances using this output format. However, you can use the **-l** option to give an exhaustive listing of information contained in the **UPS** database about the requested product.

If \$PRODUCTS contains multiple databases, output is returned for each one and labelled accordingly, for example:

```
% ups list xemacs
      DATABASE=/usr/upsII/ups_database/devel
      DATABASE=/usr/upsII/ups_database/declared/oss
      Product=xemacs Version=v19_14 Flavor=SunOS+5
      Qualifiers="" Chain=current

      DATABASE=/usr/upsII/ups_database/declared/afs
      Product=xemacs Version=v19_14 Flavor=SunOS+5
      Qualifiers="" Chain=current
```

Notice that the first database is listed in the output even though it doesn't contain any instances of the product that match the request.

Condensed Output Style

The other output format is a script-readable (or *condensed*) format, provided to allow parsing by a subsequent command. Use the **-K** option to request output in the condensed format. The **-K** option requires an argument list specifying which fields to include in the output, for example:

```
% ups list -K product:version:flavor xemacs
```

```
"xemacs" "v19_14" "SunOS+5"
```

The plus sign (+) argument, e.g., **-K+**, is a shorthand for requesting the default fields **product:version:flavor:qualifiers:chain**, for example:

```
% ups list -K+ xemacs
```

```
"xemacs" "v19_14" "SunOS+5" "" "current"
```

Some common keyword arguments used with the **-K** option are:

PRODUCT	product name
FLAVOR	product instance flavor
VERSION	product version
QUALIFIERS	additional instance specification information often used to indicate compilation options used by developer
CHAIN	product instance chain
+	all of the above
DATABASE (or DB)	the UPS database path; useful if more than one on system
DECLARER	login id of person who declared the instance
DECLARED	date/time that product instance was declared
MODIFIER	login id of person who modified/updated the instance
MODIFIED	date/time that product instance was modified/updated

If \$PRODUCTS contains multiple databases, output is returned for selected products in all of them. However, the database is identified for each output line only if the keyword DATABASE or DB is included in the argument string (e.g., **-K+ :DB** requests the standard output fields followed by the database path).

A few of the keywords allow the “at” symbol, @ to be prepended, which provides a sort of shorthand for long path names:

@PROD_DIR	entire path for the directory where the product is installed (usually equivalent to PROD_DIR_PREFIX/PROD_DIR)
@TABLE_FILE	entire path for the table file
@UPS_DIR	product’s ups directory; if it is not an absolute path, then it is taken relative to @PROD_DIR

The full list of keywords that can be used with `ups list -K` and `upd list -K` follows, with descriptions:

Keyword	Description
ARCHIVE_FILE	archive file name/location; useful with <code>upd list</code>
AUTHORIZED_NODES	authorized nodes; “all nodes” represented by an asterisk (*) in output
CATMAN_SOURCE_DIR	location of catman files (formatted man page files) included with instance
CATMAN_TARGET_DIR	directory into which catman files are to be copied
CHAIN	chain name
COMPILE_DIR	directory in which the compile file resides
COMPILE_FILE	the name of the file containing compiled functions (see Chapter 37: <i>Use of Compile Scripts in Table Files</i>)
@COMPILE_FILE	entire path to the file containing compiled functions
DECLARED	the date/time that the instance was declared to UPS or declared with a chain Note: often has multiple values, one for each declaration (e.g., for subsequent chain declarations)
DECLARER	userid of user that performed the declaration Note: often has multiple values, one for each declaration (e.g., for subsequent chain declarations)
DESCRIPTION	product description
FLAVOR	product instance flavor
HTML_SOURCE_DIR	location of html files included with instance <i>not supported in UPS v4</i>
HTML_TARGET_DIR	directory into which html files are to be copied <i>not supported in UPS v4</i>
INFO_SOURCE_DIR	location of Info files included with instance
INFO_TARGET_DIR	directory into which Info files are to be copied
MAN_SOURCE_DIR	location of unformatted man page files included with instance
MAN_TARGET_DIR	directory into which formatted man pages are to be copied
MODIFIED	last time the associated instance was changed Note: often has multiple values, one for each declaration/modification (e.g., for subsequent chain declarations)
MODIFIER	userid of user that modified the instance Note: often has multiple values, one for each declaration/modification (e.g., for subsequent chain declarations)

Keyword	Description
NEWS_SOURCE_DIR	location of news files included with instance <i>not supported in UPS v4</i>
NEWS_TARGET_DIR	directory into which news files are to be copied (for posting to a newsgroup) <i>not supported in UPS v4</i>
ORIGIN	master source file; see option -D in Chapter 24: <i>Generic Command Option Descriptions</i>
PRODUCT	product name
PROD_DIR	product root directory (usually defined relative to PROD_DIR_PREFIX, below)
@PROD_DIR	entire path to product root directory
PROD_DIR_PREFIX	product root directory prefix (area where all or most product instances are maintained)
QUALIFIERS	additional instance specification information often used to indicate compilation options used by developer
SETUPS_DIR	location of <code>setups.[c]sh</code> files and other UPS initialization files
STATISTICS	flag to record statistics for specified products
TABLE_DIR	location of table file
TABLE_FILE	name of table file
@TABLE_FILE	entire path for the table file.
UPD_USERCODE_DB	Database in which UPD configuration files are maintained (set internally by UPD ; gets its value from the UPD_USERCODE_DIR setting, with the <code>.updfiles</code> subdirectory stripped off)
UPD_USERCODE_DIR	Directory where UPD configuration files are maintained
UPS_DIR	location of <code>ups</code> directory (if not absolute path, then taken relative to PROD_DIR, usually)
@UPS_DIR	entire path to <code>ups</code> directory
VERSION	product version

22.11.5 ups list Examples

List all current products

```
% ups list
```

The simplest way to request a listing of all the current products installed in your default **UPS** database is to use the **ups list** command with no options or arguments. The per-product output spans a few lines, though, and can be cumbersome, e.g.,:

```
DATABASE=/afs/fnal.gov/ups/db
  Product=admintools   Version=v1       Flavor=SunOS+5
    Qualifiers=""      Chain=current

  Product=afsemu   Version=v1_2   Flavor=NULL
    Qualifiers=""   Chain=current

  ...

  Product=xpdf     Version=v0_7   Flavor=SunOS+5
    Qualifiers=""   Chain=current

  Product=zephyr   Version=v2_0_4 Flavor=SunOS+5
    Qualifiers=""   Chain=current
```

Use of the **-K** option with the **+** argument, e.g.,

```
% ups list -K+
```

provides the same information as the previous example, but condensed to one line per product:

```
"admintools" "v1" "SunOS+5" "" "current"
"afsemu" "v1_2" "NULL" "" "current"
...
"xpdf" "v0_7" "SunOS+5" "" "current"
"zephyr" "v2_0_4" "SunOS+5" "" "current"
```

Instead of using the **+** argument to get the default fields, you can specify particular fields:

```
% ups list -K product:version
```

This command outputs a list of product names and version numbers for all the current products installed in your default **UPS** database, e.g.,:

```
"admintools" "v1"
"afsemu" "v1_2"
...
"xpdf" "v0_7"
"zephyr" "v2_0_4"
```

List standard information for default instance of product

```
% ups list emacs
```

This command requests the standard output fields for the default instance of **emacs**, using the formatted output:

```
DATABASE=/afs/fnal.gov/ups/db
  Product=emacs   Version=v19_34b Flavor=SunOS+5
    Qualifiers=""   Chain=current
```

Addition of the **-K+** construction, e.g.,

```
% ups list -K+ emacs
```

requests the same information as the previous example, but in condensed format:

```
"emacs" "v19_34b" "SunOS+5" " " "current"
```

Using **-a** for all, e.g.,

List standard information for all instances of product

```
% ups list -a emacs
```

requests the standard output fields for all instances of **emacs**, using the formatted output:

```
DATABASE=/afs/fnal.gov/ups/db
  Product=emacs  Version=v19_30a Flavor=AIX+3
    Qualifiers=""  Chain=""

  Product=emacs  Version=v19_30a Flavor=IRIX+5
    Qualifiers=""  Chain=""

  Product=emacs  Version=v19_30a Flavor=SunOS+5
    Qualifiers=""  Chain=""
...

  Product=emacs  Version=v19_34b Flavor=SunOS+5
    Qualifiers=""  Chain=current
```

Use of the **-K** option with the **+** argument, e.g.,

```
% ups list -aK+ emacs
```

requests the same information as the previous example, but in condensed format (**-K+**):

```
"emacs" "v19_30a" "AIX+3" " " " "
"emacs" "v19_30a" "IRIX+5" " " " "
"emacs" "v19_30a" "SunOS+5" " " " "
...
"emacs" "v19_34b" "SunOS+5" " " "current"
```

List standard information for all instances of product for your machine's flavor

```
% ups list -a2K+ emacs
```

To request information on all instances of a product limited to the OS of your machine, you can include the **-f** option (for flavor), or enter a command like this one, where **-a** is for all, **-2** is for the “level 2” designation of your machine's OS, and **-K+** is for condensed output:

```
"emacs" "v19_30a" "SunOS+5" " " " "
"emacs" "v19_34" "SunOS+5" " " " "
"emacs" "v19_34b" "SunOS+5" " " "current"
```

List specific keywords for a product

If your installation has multiple databases defined in \$PRODUCTS, it is useful to include the keyword for the database (DB) in the **-K** argument list, e.g.,

```
% ups list -aK+:DB emacs
```

This example is similar to the previous one, but the database path is included at the end:

```
"emacs" "v19_30a" "AIX+3" " " " " "/afs/fnal.gov/ups/db"
"emacs" "v19_30a" "IRIX+5" " " " " "/afs/fnal.gov/ups/db"
"emacs" "v19_30a" "AIX+3" " " " " "/usr/products/ups_database/main"
"emacs" "v19_30a" "IRIX+5" " " " " "/usr/products/ups_database/main"
...
```

```
% ups list -K product:version admintools
```

This specifies particular fields to be output with the **-K** option for the default instance of the product **admintools**, producing:

```
"admintools" "v1"
```

List all keywords for a product (long listing)

```
% ups list emacs -l
```

This command requests detailed information (**-l** for long listing) about the default instance of the product **emacs**. Administrative users may often need this level of detailed information about a product. The **-l** option is not valid with the **-K** option. The output looks like this:

```
DATABASE=/afs/fnal.gov/ups/db
Product=emacs Version=v19_34b Flavor=SunOS+5
Qualifiers="" Chain=current
Declared="1997-07-15 00.00.00 GMT:1997-03-21 00.00.00 GMT"
Declarer="root:colossus"
Modified=":1997-03-21 00.00.00 GMT"
Modifier=":colossus"
Home=/afs/fnal.gov/products/SunOS+5/emacs/v19_34b
No Compile Directive
Authorized, Nodes=*
UPS_Dir="ups"
Table_Dir=""
Table_File="v19_34b.table"
Archive_File=""
Description=""
Action=setup
    proddir()
    setupenv()
    sourceRequired(${UPS_UPS_DIR}/setup.${UPS_SHELL},UPS_ENV)
Action=Configure
    ProdDir()
    execute(${UPS_UPS_DIR}/configure,UPS_ENV)
    UnProdDir()
Action=unConfigure
    ProdDir()
    execute(${UPS_UPS_DIR}/unconfigure,UPS_ENV)
    UnProdDir()
Action=Current
    ProdDir()
    execute(${UPS_UPS_DIR}/configure,UPS_ENV)
    execute(${UPS_UPS_DIR}/current,UPS_ENV)
    UnProdDir()
```

```

Action=unCurrent
  ProdDir()
  execute(${UPS_UPS_DIR}/uncurrent,UPS_ENV)
  UnProdDir()

```

This gives a fairly long list. It is often more convenient to use the **-K** option with a list of keywords for the specific fields you need.

Use “ups list -K” to locate product root directory, table file and ups directory

ups list -K can be used to locate a product’s root directory, table file, and **ups** directory when used with the keywords corresponding to these quantities

Compare the following three commands and their output. **UPS_DIR** represents the location of the product’s **ups** directory. If it is not an absolute path, then it is taken relative to **@PROD_DIR**, if specified (as shown in the second command). **@UPS_DIR** is the absolute path.

```
% ups list -K @PROD_DIR teledata
```

```
"/afs/fnal.gov/ups/teledata/v1_0/NULL"
```

```
% ups list -KUPS_DIR teledata
```

```
"ups"
```

```
% ups list -K@UPS_DIR teledata
```

```
"/afs/fnal.gov/ups/db/teledata/v1_0/NULL/ups"
```

Compare the following two commands and their output. **table_file** represents only the name of the table file, not its path. **@table_file** is the entire path for the table file. See section 28.4 *Determination of ups Directory and Table File Locations* for information on how **UPS** determines the table file directory.

```
% ups list -Ktable_file teledata
```

```
"v1_0.table"
```

```
% ups list -K@table_file teledata
```

```
"/afs/fnal.gov/ups/db/teledata/v1_0.table"
```

Parse output from “ups list -K” in perl

Here we provide guidance on parsing **ups list** output in **perl**, with all the appropriate quoting and spacing. The following example first defines the file handle **UPS_LIST_OUTPUT** to contain the piped output of the command **\$UPS_DIR/bin/ups list -aK+** (where **\$UPS_DIR** is translated inside of **perl** via the **\$ENV{UPS_DIR}** statement, which is the translation of the environmental variable **UPS_DIR**). It then defines the array **@fields**, and parses the output into a set of five variables.

```

open(UPS_LIST_OUTPUT, "| $ENV{UPS_DIR}/bin/ups list -aK+");
while (<UPS_LIST_OUTPUT> ) {
  # break into the array @fields:
  @fields = m/"(?:[^\\"\\]|\\.)*"/g;
  # then do things with $field[0] $field[1] ...
  # (in this case,
  # $field[0] = product name
  # $field[1] = version
  # $field[2] = flavor
  # $field[3] = qualifiers (colon-separated list)
  # $field[4] = chains (colon-separated list)
  }

```

Say the output from your **ups list -K** command looks like this:

```
"ups" "v4_4" "IRIX+5" "" "current"
```

then the `@fields` array contains the variables:

```
$field[0] = ups
$field[1] = v4_4
$field[2] = IRIX+5
$field[3] =
$field[4] = current
```

Alternatively, you could parse the output this way:

```
($product, $version, $flavor, $qualifiers, $chains) = m/\("(?:[^\\"|\\\.]*)"\)/g;
```

and then you'd have:

```
$product = ups
$version = v4_4
$flavor = IRIX+5
$qualifiers =
$chains = current
```

Parsing output from “ups list -K” in a sh script

You can parse the output from a command of the form **ups list -K** by piping it into a “*while* loop” **sh** script. Here is an example; explanations follow the code:

```
ups list -K... |
while read line
do
    eval set : $line
    shift
    # now do things with $1 $2 $3...
done
```

As the condition of the *while* loop, the **read line** command reads the lines of output into the variable *line*. To get rid of the quotes, the loop runs **eval set : \$line** on each line (this syntax ensures that the **set** command actually sets the variables `$1`, `$2`, and so on, instead of setting shell behavior in case the first argument starts with a dash). The **shift** that follows then gets rid of the colon.

22.12 ups modify

The `ups modify` command allows you to manually edit any of the database product files. It performs syntax and content validation before and after the editing session.

22.12.1 Command Syntax

```
% ups modify [<options>] <product> [<version>]
```

22.12.2 Commonly Used Options

See section 22.12.3 *All Valid Options* for descriptions of each option.

```
-E <editor>
-f <flavor>          Or one of -0, -1, -2, -3, or -H (alone or together with one of
                    -0, -1, -2, -3)
-g <chainName>      Or one of -c, -d, -n, -o, -t
-N <fileName>
-q <qualifierList>
-z <databaseList>
```

22.12.3 All Valid Options

```
-? ("-?" for csh)  Prints command description and option usage information to screen
-a                Operates on all instances that match the other options given on
                  command line
-E <editor>       Invokes the specified editor. If not given, uses the editor specified by
                  $EDITOR. If not set, uses vi by default.
-f <flavor>       Finds product instance of specified flavor. If specified and no exact
                  match is found, the command fails. Multiple values accepted, but UPS
                  looks only at first in list.
-H <flavor>       Specifies flavor and builds a flavor list for that family starting at the
                  level specified. UPS finds the best match instance for the specified
                  flavor family. Multiple values accepted, but UPS looks only at first in
                  list.
-m <tableFileName> Specifies table file name
-M <tableFileDir> Specifies table file directory
-N <fileName>     Specifies file to be checked and edited
-p "<description>" Specifies product description
```

- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- T <path or URL>** Specifies archive file path or URL. This is used only for declarations in distribution databases for which products are maintained in tar or gzip (archived) format.
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product
- Z** Times the command

22.12.4 More Detailed Description

`ups modify` performs the following steps (if you specify the file using `-N`, the menu will not appear):

- presents menu of files that you can edit and asks you to either select one or quit
- verifies pre-modification contents of file (runs `ups verify`)
- starts up the editor given by `-E <editor>` or, if that is not specified, then `$EDITOR`, if set. If neither is specified, it starts up `vi` by default.
- makes a copy of the file to be edited
- pulls copy of file into the editor
- after user exits the editor, runs `ups verify` on the edited file
- if the validation succeeds, writes the new file over the old one and quits
- if the validation does not succeed, provides informational messages, and quits
- if no changes made to file, again presents menu of files

Internal Processes

- Bring up requested file in specified editor
- Verify the pre-edited file
- Verify the edited file before overwriting
- Process MODIFY action
- Execute the temp file

22.12.5 ups modify Example

```
% ups modify teledata v1_0 -N $MYDB/teledata/v1_0.version
```

In this example, we select the version file (via **-N**) for the product **teledata** v1_0 (default flavor, no qualifiers). Since **-E** is not given, **UPS** will use the editor set in **\$EDITOR**, or **vi** if that variable is not set. First, **UPS** runs **ups verify** and produces the output:

```
Pre modification verification pass complete.
```

No errors were detected. The version file is next displayed in the editor.

1) To illustrate an unsuccessful validation, we add a bogus line:

```
TESTKEYWORD = value
```

and save and quit. **UPS** returns the following messages, and we opt to save the erroneous change:

```
INFORMATIONAL: Unexpected key word 'TESTKEYWORD' in '/home/t1/ahavey/upsII/declared/teledata/v1_0.version', line 17
```

```
INFORMATIONAL: Unexpected key word 'TESTKEYWORD' in '/home/t1/ahavey/upsII/declared/teledata/v1_0.version', line 17
```

```
Post modification verification pass complete.
```

```
Do you wish to save this modification [y/n] ? y
```

UPS quits, saving the file as we requested.

2) To illustrate successful validation, we'll correct the error introduced above. We run the same **ups modify** command. **UPS** finds the error during the pre-edit validation:

```
INFORMATIONAL: Unexpected key word 'TESTKEYWORD' in '/home/t1/ahavey/upsII/declared/teledata/v1_0.version', line 17
```

```
INFORMATIONAL: Unexpected key word 'TESTKEYWORD' in '/home/t1/ahavey/upsII/declared/teledata/v1_0.version', line 17
```

```
Pre modification verification pass complete.
```

We remove the incorrect line from the version file, then save and quit. **UPS** displays the following message, and we elect to save the change (**y**):

```
Post modification verification pass complete.
```

```
Do you wish to save this modification [y/n] ? y
```

UPS quits, saving the file as requested.

22.13 ups start

The **ups start** command is used to invoke appropriately configured products at system boot time. This is used primarily for products that need to load drivers, start daemons or perform other actions required at boot time. This command is generally not run manually, rather it gets executed from within a **UPS** control file.

The **ups stop** command (see section 22.14 *ups stop*) is used to stop products that are started this way.

22.13.1 Command Syntax

```
% ups start [<options>] <product> [<version>]
```

22.13.2 Commonly Used Options

See section 22.13.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-v(vvv)

-w

-z <databaseList>

22.13.3 All Valid Options

-? ("**-?**" for **cs**) Prints command description and option usage information to screen

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-f <flavor> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under "The flavor options".

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

-n Finds product instance chained to "new"

-o Finds product instance chained to "old"

-O "<flags>" Sets the value of **\$UPS_OPTIONS** to **<flags>**.

- P** Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- w** Stops the product first, then restarts it
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

-3

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.13.4 More Detailed Description

Please see Chapter 14: *Automatic UPS Product Startup and Shutdown* for a detailed description of the use of the **ups start** command. In brief, it covers:

- Configuring your machine to allow automatic startup/shutdown
- Installing a **UPS** product to start and/or stop automatically, for which you need to:
 - Determine if auto start/stop feature is enabled
 - Determine if product is appropriate for autostart
 - Edit control file(s)
 - Restart the system

Internal Processes

- Check node authorization
- If necessary, process STOP action
- Process START action
- Execute the temp file

22.13.5 ups start Examples

Run the command interactively

This command first stops (**-w**) the default instance of the product **apache**, and then restarts it:

```
% ups start -w apache
```

It prints to screen messages of the format:

```
Stopping Apache server for fnkits.fnal.gov on port 8000
Stopping Apache server for fnkits.fnal.gov on port 8000 account updadmin
```

Run the command from a control file

This command starts the default instance of the product **ObjectCenter** for the flavor SunOS, and requests verbose output (**-v**).

```
% ups start -v -f SunOS ObjectCenter
```

If the logon id is *root*, the line in the control file may look like:

```
ups start -v -f SunOS ObjectCenter
```

If the logon id is other than *root*, the line in the control file must look like:

```
/bin/su - products -c "ups start -v -f SunOS ObjectCenter"
```


22.14 ups stop

The **ups stop** command is used to stop appropriately configured products at system shutdown. This command is generally not run manually, rather it gets executed from within a **UPS** control file, and operates on products that were invoked using **ups start** (see section 22.13 *ups start*). Typically, these products are ones which need to load drivers, start daemons or perform other actions required at boot time.

22.14.1 Command Syntax

```
% ups stop [<options>] <product> [<version>]
```

22.14.2 Commonly Used Options

See section 22.14.3 *All Valid Options* for descriptions of each option.

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (alone or together with one of  
                      -0, -1, -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-q <qualifierList>  
-v(vvv)  
-z <databaseList>
```

22.14.3 All Valid Options

```
-? ("-?" for csh)    Prints command description and option usage information to screen  
-c                  Finds product instance chained to "current"  
-d                  Finds product instance chained to "development"  
-f <flavor>         Described below under "The flavor options".  
-g <chainName>      Finds product instance chained to <chainName>  
-H <flavor>         Described below under "The flavor options".  
-m <tableFileName> Specifies table file name  
-M <tableFileDir>  Specifies table file directory  
-n                  Finds product instance chained to "new"  
-o                  Finds product instance chained to "old"  
-O "<flags>"        Sets the value of $UPS_OPTIONS to <flags>.
```

- P** Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

-3

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.14.4 More Detailed Description

Please see Chapter 14: *Automatic UPS Product Startup and Shutdown* for a detailed description of the use of the **ups stop** command. In brief, it covers:

- Configuring your machine to allow automatic startup/shutdown
- Installing a **UPS** product to start and/or stop automatically, for which you need to:
 - Determine if auto start/stop feature is enabled
 - Determine if product is appropriate for autostart
 - Edit control file(s)
 - Restart the system

Internal Processes

- Check node authorization
- Process STOP action
- Execute the temp file

22.14.5 ups stop Examples

Run the command interactively

This command stops the default instance of the product **apache**:

```
% ups stop apache
```

It prints to screen a message of the format:

```
Stopping Apache server for fnkits.fnal.gov on port 8000
```

Run the command from a control file

This command stops the default instance of the product **ObjectCenter** for the flavor SunOS, and requests verbose output (**-v**).

```
% ups stop -v -f SunOS ObjectCenter
```

If the logon id is *root*, the line in the control file may look like:

```
ups stop -v -f SunOS ObjectCenter
```

If the logon id is other than *root*, the line in the control file must look like:

```
/bin/su - products -c "ups stop -v -f SunOS ObjectCenter"
```


22.15 ups tailor

For any product instance whose table file includes a TAILOR action, the **ups tailor** command must be run manually at product installation time after the product is declared to the database in order to execute this action. A TAILOR action typically includes installation functions which require input from the installer.

22.15.1 Command Syntax

```
% ups tailor [<options>] <product> [<version>]
```

22.15.2 Commonly Used Options

See section 22.15.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-v(vvv)

-z <databaseList>

22.15.3 All Valid Options

-? (" -?" for csh) Prints command description and option usage information to screen

-c Finds product instance chained to “current”

-d Finds product instance chained to “development”

-f <flavor> Described below under “The flavor options”.

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under “The flavor options”.

-K <keywordList> Returns values of specified keywords only; valid keywords are listed in section 27.4 *List of Supported Keywords*

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

-n Finds product instance chained to “new”

-o Finds product instance chained to “old”

-O "<flags>" Sets the value of \$UPS_OPTIONS to **<flags>**.

- P** Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)
- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of **-0**, **-1**, **-2**, **-3**, **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.15.4 More Detailed Description

Tailoring is the aspect of the product implementation that requires input from the product installer (e.g., specifying the location of hardware devices for a software driver package). If the product requires tailoring, a file is usually supplied in the format of an interactive executable (script or compiled binary), and it is run via a function under the ACTION=TAILOR line in the table file. Configure, tailor and other supplemental installation scripts are usually maintained in the product's `$(PRODUCT)_DIR/ups` directory. You must explicitly tailor the product instance using the UPS command **ups tailor**; tailoring is *not* performed automatically.

Tailoring is generally allowed on any node of a cluster, however we strongly recommend that you perform any node-specific tailoring from that node, or flavor-specific tailoring from a node of that flavor to avoid mismatches.

Internal Processes

- Check node authorization
- Process TAILOR action
- Execute the temp file

22.15.5 ups tailor Example

As an example, we show the tailor process for the product **apache**:

```
% ups tailor apache

Current configuration nicknames are:
kits    kits8k
You can:
  a)dd a new server configuration
  q)uit the tailor script
Which would you like? [aq]? a
Webserver alias/name (e.g. www-xx.fnal.gov)? www-demo.fnal.gov
Webserver nickname [demo]?
Webserver port number [80]?
Webserver effective user-id [wwwsrv]?
Webserver effective group-id [www]?
Webserver admin id [wwwadm]?
Mail address for admin stuff [mengel@fnal.gov]? demo-admin@fnal.gov
Directory for CGI executables [/fnal/www/demo/cgi-bin]?
Directory /fnal/www/demo/cgi-bin doesn't exist, make it [y]? y
Root of served files [/fnal/www/demo/html]?
Directory /fnal/www/demo/html doesn't exist, make it [y]? y
Raw Log file directory [/var/adm/www/demo]? /tmp/demo
Log file Summary directory [/fnal/www/demo/html/logs]?
Directory /fnal/www/demo/html/logs doesn't exist, make it [y]? y
Current configuration nicknames are:
demo    kits    kits8k
```

You can:
a)dd a new server configuration
q)uit the tailor script
Which would you like? [aq]? q

22.16 ups touch

The **ups touch** command changes the values of the keywords **MODIFIED** and **MODIFIER** in the version file to the current time and the current user, respectively.

If you make any changes to a product's database files by hand (e.g., not via **ups modify**), you should run **ups touch** afterwards.¹ You can also run it to prevent an update if you choose to run **upd update** on several product instances at a time.

22.16.1 Command Syntax

```
% ups touch [<options>] <product> [<version>]
```

22.16.2 Commonly Used Options

See section 22.16.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-z <databaseList>

22.16.3 All Valid Options

-? ("**-?**" for **cs**) Prints command description and option usage information to screen

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-f <flavor> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under "The flavor options".

-n Finds product instance chained to "new"

-o Finds product instance chained to "old"

-q <qualifierList> Finds product instance with the specified qualifiers (required and/or optional)

-t Finds product instance chained to "test"



1. We recommend that you use **ups modify** to edit the database files, in which case you don't need to run this command. **ups modify** also runs **ups verify** to prevent entry of database errors.

- v(vvv)** Prints out extra debugging information.
- z <databases>** Specifies the database(s) to use
- Z** Times the command (does not include time for sourcing of temp file for **setup/unsetup**)

The flavor options

Flavor may be specified using **-f**, using **-H** in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Must be used with any of **-0**, **-1**, **-2**, **-3**. Specifies flavor and builds a flavor list for that family starting at the level specified. **UPS** finds the product instance of specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

22.16.4 ups touch Example

To illustrate this command, we first run a **ups list** showing the modified date/time and the modifier(s) for a product:

```
% ups list teledata -aKproduct:modified:modifier
"teledata" "1999-09-08 21.44.04 GMT:1999-09-08 21.38.23 GMT" "olduser:olduser"
```

Now run **ups touch** to change these values:

```
% ups touch teledata
```

And verify that they have changed, by rerunning the **ups list** command:

```
% ups list teledata -aKproduct:modified:modifier
"teledata" "1999-11-19 18.43.00 GMT:1999-09-08 21.38.23 GMT" newuser:olduser"
```

22.17 ups unconfigure

For any product instance whose table file includes an UNCONFIGURE action, the **ups unconfigure** command executes this action. An UNCONFIGURE action usually includes functions that reverse, approximately or fully, the functions run by the CONFIGURE action. The **ups unconfigure** command gets run by default by **ups undeclare** when the product is removed from a database (see section 22.18 *ups undeclare*, in particular the **-C** option), but can be run manually as needed.

22.17.1 Command Syntax

```
% ups unconfigure [<options>] <product> [<version>]
```

22.17.2 Commonly Used Options

See section 22.17.3 *All Valid Options* for descriptions of each option.

-f <flavor> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-z <databaseList>

22.17.3 All Valid Options

-? ("**-?**" for **cs**) Prints command description and option usage information to screen

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-f <flavor> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavor> Described below under "The flavor options".

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

-n Finds product instance chained to "new"

-o Finds product instance chained to "old"

-O "<flags>" Sets the value of **\$UPS_OPTIONS** to **<flags>**.

-P Requires **UPS** to rely only on information supplied on the command line to locate the product instance (prevents **UPS** from searching in a database)

- q <qualifierList>** Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavor>** Finds product instance of specified flavor. If specified and no exact match is found, the command fails. Multiple values accepted, but **UPS** looks only at first in list.
- H <flavor>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted, but **UPS** looks only at first in list.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of `-0`, `-1`, `-2`, `-3`, **UPS** finds the product instance of specified level of that flavor; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.2.

22.17.4 More Detailed Description

A product's configuration may involve creating links to the product root directory from other areas (see section 15.1.3 *Third-Party Products Requiring a Hard-Coded Path*). If the area is not identical for each node accessing the UPS database in which the product instance has been declared (i.e., same path but separate areas), then the **ups configure** command needs to be run manually on each node that mounts a unique area. Similarly, when removing such a product from a database, you will need to run the **ups unconfigure** command manually on each node. If you are not sure whether you need to unconfigure a product instance on each node, look through the ACTION=UNCONFIGURE steps in the table file to see what they do.

Internal Processes

- Check node authorization
- Process the UNCONFIGURE action
- Execute the temp file

22.17.5 ups unconfigure Example

perl is a product that requires **ups configure** and **ups unconfigure** to be run manually for each machine flavor in a cluster. The sample command, which should be issued from a machine of flavor SunOS+5, runs the UNCONFIGURE action in the table file associated with the product **perl**, version v5_005 for flavor SunOS+5. If that action is not present, it undoes all the reversible functions included under the CONFIGURE action, by default.

```
% ups unconfigure perl v5_005 -f SunOS+5
```

This command should take care of the “unconfiguration” of **perl** on all the machines of flavor SunOS+5 in the cluster. A command like this, but with the appropriate flavor, must be run for each machine flavor represented in the cluster.

22.18 ups undeclare

The **ups undeclare** command is used for two separate purposes:

- 1) to remove an instance from a database (and optionally remove its product root directory); the information that gets removed includes:
 - the version file, or the portion of the version file, that pertains to the instance
 - any chain files, or the portions of any chain files, that pertain to the instance
- 2) to remove a chain from an instance

22.18.1 Command Syntax

For removing an instance

```
% ups undeclare <flavor_option> [<other_options>] <product> \  
  <version>
```

For removing a chain

```
% ups undeclare <chain_option> [<other_options>] <product>
```

22.18.2 Commonly Used Options

See section 22.17.3 *All Valid Options* for descriptions of each option.

For removing an instance

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                      -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-q <qualifierList>  
-Y  
-Y  
-z <databaseList>
```

For removing a chain

```
-f <flavor>           Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                      -2, -3)  
-g <chainName>       Or one of -c, -d, -n, -o, -t  
-q <qualifierList>  
-z <databaseList>
```

22.18.3 All Valid Options

Valid only for removing an instance (not for removing a chain)

- y Deletes product root directory, provides confirmation prompt
- Y Deletes product root directory, does not provide confirmation prompt

Valid for both functions

- ? ("**-?**" for **cs**) Prints command description and option usage information to screen
- c Finds product instance chained to "current"
- C When removing a product: Prevents execution of the UNCONFIGURE action
When removing a chain: Prevents execution of the corresponding "unchain" action
- d Finds product instance chained to "development"
- f <flavor> Described below under "The flavor options".
- g <chainName> Finds product instance chained to <chainName>
- H <flavor> Described below under "The flavor options".
- m <tableFileName> Specifies table file name
- M <tableFileDir> Specifies table file directory
- n Finds product instance chained to "new"
- o Finds product instance chained to "old"
- O "<flags>" Sets the value of \$UPS_OPTIONS to <flags>.
- q <qualifierList> Finds product instance with the specified qualifiers (required and/or optional)
- r <prodRootDir> Specifies the product root directory
- t Finds product instance chained to "test"
- U <upsDir> Specifies location of ups directory; default value is ups
- v(<vvv>) Prints out extra debugging information.
- V Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databases> Specifies the database(s) to use
- Z Times the command

The flavor options

Flavor may be specified using `-f`, using `-H` in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- | | |
|--------------------------------|--|
| <code>-f <flavor></code> | Finds product instance of specified flavor. If specified and no exact match is found, the command fails. |
| <code>-H <flavor></code> | Must be used with any of <code>-0</code> , <code>-1</code> , <code>-2</code> , <code>-3</code> . Specifies flavor and builds a flavor list for that family starting at the level specified. UPS finds the product instance of specified level of that flavor; e.g., <code>-2H IRIX+6.2</code> is equivalent to <code>-f IRIX+6</code> . |
| <code>-0</code> | Specifies flavor as NULL; equivalent to <code>-f NULL</code> |
| <code>-1</code> | Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to <code>-f SunOS</code> ; if given together with <code>-H IRIX+6.2</code> , flavor is then specified as IRIX. |
| <code>-2</code> | Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to <code>-f SunOS+5</code> ; if given together with <code>-H IRIX+6.2</code> , flavor is then specified as IRIX+6. |
| <code>-3</code> | Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to <code>-f SunOS+5.6</code> ; if given together with <code>-H IRIX+6.2</code> , flavor is then specified as IRIX+6.2. |

22.18.4 More Detailed Description

Removing a Product Instance



Using `ups undeclare` is the recommended procedure for removing product instances. Removing them manually does not ensure that all the files get deleted or that chains get updated properly, which can lead to a fragmented products area.

To undeclare a product instance, you must specify the *version* of the instance, not its *chain*, in the `ups undeclare` command. **Specifying the chain removes only that chain, not the instance itself.** When an instance gets “undeclared”, all information pertaining to it is removed from the **UPS** database in question; this includes:

- the version file, or the portion of the version file, that pertains to the instance
- any chain files, or the portions of any chain files, that pertain to the instance

You can also opt to remove the product instance’s directory tree starting from its root directory. To do so, use one of the options `-y` or `-Y` (`-y` queries you for confirmation, `-Y` does not).

Before removing anything, you should find out if any other products have the product instance in question declared as a dependency.¹ If so, you may want to reconsider removing it. Removal of the product instance may affect the operation of its parent products.

1. The `ups parent` command will provide this information. The command is not available in **UPS** version v4; it is planned for a future release.

We recommend always including a flavor option if you have a multi-flavor database.

The **ups undeclare** command executes **ups unconfigure** by default (the UNCONFIGURE process can be suppressed by using the **-C** option, however normally you want this process to execute).



Special case: If a product has a CONFIGURE action that modifies files outside of its product root directory, and if this instance is used by more than one node, flavor or file system, then you may need to run **ups undeclare** or **ups unconfigure** on all of the nodes before removing the product files on any node. Check the product's table file.

Removing a Chain from an Instance

To remove a chain, include the chain specification in the **ups undeclare** command, but do not include the version. Including both the chain and version is bound to be either redundant or incompatible, and may result in removing the product declaration! We recommend always including the **-f <flavor>** option if you have a multi-flavored database.

Internal Processes

- Find database to use
- If necessary process all appropriate 'UNCHAIN' actions
- Process the UNCONFIGURE action
- Process the UNDECLARE action
- If necessary delete the product's home area
- Execute the temp file

22.18.5 ups undeclare Examples

Undeclare an instance

```
% ups undeclare -f IRIX+5 tcl v7_6a -y
```

In this example, we undeclare and remove current instance (by default) of the product **tcl v7_6a** for the flavor **IRIX+5** (**-f** option) from the default database. Notice that the version is included for instance identification as required. We opt to remove the product root directory after query (lowercase **-y** option):

```
Product home directory -  
    /export/home/t1/ahavey/upsII/products/tcl/v7_6a/  
Delete this directory?y
```

We respond "y" for yes. To respond no, we would enter "n".

Undeclare a chain

```
% ups undeclare -c ximagetools -f NULL
```

In this command, we remove the "current" chain (**-c** option) from the instance of **ximagetools** declared as current for the flavor **NULL** (**-f** option). Notice that the version is not included!



If multiple flavor/qualifier pairs share the chain file in question (in which case you need to specify the flavor/qualifier information on the command line), only the portion of the chain file pertaining to the specified instance will get removed; the file itself will not be deleted.

22.19 ups verify

The **ups verify** command checks the integrity of the database files for the specified product(s), and lists any errors and inconsistencies that it finds.

The **ups verify** command gets run by **ups modify** before and after file editing (see section 22.12 *ups modify*). **ups verify** can also be run manually as needed.

22.19.1 Command Syntax

```
% ups verify [<options>] [<product>] [<version>]
```

22.19.2 Commonly Used Options

See section 22.19.3 *All Valid Options* for descriptions of each option.

-f <flavorList> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)

-g <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**

-q <qualifierList>

-z <databaseList>

22.19.3 All Valid Options

-? ("**-?**" for **csH**) Prints command description and option usage information to screen

-a Verifies files for all instances that match the other options given on command line

-c Finds product instance chained to "current"

-d Finds product instance chained to "development"

-f <flavorList> Described below under "The flavor options".

-g <chainName> Finds product instance chained to **<chainName>**

-H <flavorList> Described below under "The flavor options".

-m <tableFileName> Specifies table file name

-M <tableFileDir> Specifies table file directory

-n Finds product instance chained to "new"

-o Finds product instance chained to "old"

-q <qualifierList> Finds product instance with the specified qualifiers (required and/or optional)

-r <prodRootDir> Specifies the product root directory

- t** Finds product instance chained to “test”
- U <upsDir>** Specifies location of `ups` directory; default value is `ups`
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the database(s) in which to look for the product and its dependencies
- Z** Times the command (does not include time for sourcing of temp file for `setup/unsetup`)

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavorList>** Finds product instance of specified flavor(s). If specified and no exact match is found, the command fails. Multiple values accepted.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted.

Can be used alone (without an accompanying number option). In this case, **UPS** finds the best match instance for the specified flavor family. If used with any of `-0`, `-1`, `-2`, `-3`, **UPS** finds the product instance of specified level of that flavor; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.2.

22.19.4 ups verify Example

```
% ups verify -z $MYDB teledata
```

For this example, we have given an erroneous value to the TABLE_FILE keyword in the version file for this product. The command output shows:

```
ERROR: No instance matches were made between the
version file (/home/t1/aheavey/upsII/declared/teledata/v1_0.version) and the
table file (v1_1.table) for flavor (NULL) and qualifiers ()
ERROR: Possible UPS database (/home/t1/aheavey/upsII/declared) corruption in pro
duct 'teledata'.
ERROR: No instance matches were made between the chain file (/home/t1/aheavey/up
sII/declared/teledata/current.chain) and the version file (v1_0.version)
ERROR: Possible UPS database (/home/t1/aheavey/upsII/declared) corruption in pro
duct 'teledata'.
```


Chapter 23: UPD/UPP Command Reference

This chapter contains full usage information on all the **UPD** commands and the **UPP** command. In particular, for each command you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a listing of commonly used options, without descriptions
- a listing of all valid options, with command-specific descriptions
- (as needed) a section called “Options Valid with -G”
- (as needed) a section called “More Detailed Description” which typically includes detailed command-specific usage information
- command examples

For commands that have a corresponding **UPS** command, you will find:

- a statement of the purpose and/or function of the command
- the command syntax
- a reference to the corresponding **UPS** command
- a listing of any additional, **UPD**-specific options
- (as needed) command examples

For the **upd addproduct** and **upd install** commands we include a detailed list of the internal processes the command performs. The internal processes for the other commands can largely be inferred from these lists.

23.1 upd addproduct

The `upd addproduct` command adds a product instance to a product distribution database. A product instance may contain any or all of the following: a product root directory, a table file, and/or a `ups` directory. The product may be in tar file format or unwound. `upd addproduct` declares the product instance on the distribution node with the same product instance identifiers (e.g., product name, version, flavor, qualifiers, chain) as it has on the local node.

23.1.1 Command Syntax

For adding a product that is declared to a local database

An unwound product or a table file:

```
% upd addproduct [<flavor_option>] [<other_options>] <product> \  
  <version>
```

A product tar file:

```
% upd addproduct [<flavor_option>] -T <archFilePath> \  
  [<other_options>] <product> <version>
```

Note: The flavor option is not strictly required; `upd addproduct` defaults to the current instance on the local system, similarly to other **UPS** and **UPD** commands.

For adding a product that is not declared to a local database

An unwound product:

```
% upd addproduct [-P] <flavor_option> -r <prodRootDir>      \  
  -m <tableFileName> [-M <tableFileDir>] [-U <upsDir>]      \  
  [<other_options>] \ <product> <version>
```

A table file:

```
% upd addproduct [-P] <flavor_option> -m <tableFileName> \  
  [-M <tableFileDir>] [<other_options>] <product> <version>
```

A product tar file:

```
% upd addproduct [-P] <flavor_option> -T <archFilePath>    \  
  -m <tableFileName> [-M <tableFileDir>] [<other_options>] \  
  <product> <version>
```

Notes:

- The `-P` option is not strictly required, however it is recommended. In cases where the local database contains a product instance that also matches the specified options, `-P` ensures that **UPD** ignores the database and picks up the intended instance.
- If the product includes a table file, you must include the `-m` option specifying its name, as there is no default. You must also include `-M` if the table file is not in the current directory.
- If the `ups` directory is in the default location (`$(PRODUCT)_DIR/ups`), **UPS** should be able to find it, but it's safer to always specify it via the `-U` option.

23.1.2 Commonly Used Options

See section 23.1.3 *All Valid Options* for descriptions of each option.

For adding a product that is declared to a local database

- f** <flavorList> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (alone or together with one of **-0**, **-1**, **-2**, **-3**)
- g** <chainName> Or one of **-c**, **-d**, **-n**, **-o**, **-t**
- h** <host>
- q** <qualifierList>
- T** <archFilePath>
- z** <databaseList>

For adding a product that is not declared to a local database

- f** <flavorList> Or one of **-0**, **-1**, **-2**, **-3**, or **-H** together with one of **-0**, **-1**, **-2**, **-3**
- h** <host>
- m** <tableFileName>
- M** <tableFileDir>
- P**
- q** <qualifierList>
- r** <prodRootDir>
- T** <archFilePath>
- z** <databaseList>

23.1.3 All Valid Options

- ?** ("**-?**" for **csH**) Prints command description and option usage information to screen
- A** <nodeList> Specifies one or more nodes on which use of product is authorized (passed through to **UPS**)
- c** Finds product instance chained to “current” on local node, and chains it to “current” on the distribution node. If **-P** used, ignores any local chain, chains selected instance to “current” on the distribution node.
- d** Finds product instance chained to “development” on local node, and chains it to “development” on the distribution node. If **-P** used, ignores any local chain, chains selected instance to “development” on the distribution node.
- D** "<origin>" Specifies the product's master source file for informational purposes. Not fully implemented; currently just passed through to **UPS**.
- f** <flavorList> Described below under “The flavor options”.



- g <chainName>** Finds product instance chained to **<chainName>** on local node, and chains it to **<chainName>** on the distribution node. Can be a list of chain names. If **-P** used, ignores any local chain, chains selected instance to “**<chainName>**” on the distribution node.

- h <host>** Specifies product distribution host to which you are adding the product; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:
 - a plain host name; we recommend using the full host name with **upd addproduct -h <host>** (e.g., *fred.fnal.gov*, rather than just *fred*) to prevent problems when people download the product to off-site user nodes. (Using *fred* by itself in **upd addproduct** is possible only if the *sub.domain* is *fnal.gov*; in other commands it is fine to use it that way.)
 - a host and Webserver port number (e.g., *fred.sub.domain:8080*)
 - a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)

- H <flavorList>** Described below under “The flavor options”.

- i** Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one, operation continues to the next.

- m <tableFileName>** Specifies table file name on local node. Required for products that are not declared locally (or when **-P** used); if product is declared locally, can be used to override the corresponding value set in the local declaration.

- M <tableFileDir>** Specifies table file directory on local node.

Generally used with products that are not declared locally (or when **-P** used). In this case, it is required whenever the table file is in a directory other than the current directory.

If product is declared locally, can be used to override the corresponding value set in the local declaration.

- n** Finds product instance chained to “new” on local node, and chains it to “new” on the distribution node. If **-P** used, ignores any local chain, chains selected instance to “new” on the distribution node.

- o** Finds product instance chained to “old” on local node, and chains it to “old” on the distribution node. If **-P** used, ignores any local chain, and chains selected instance to “old” on the distribution node.

- O "<flags>"** Sets the value of `$UPS_OPTIONS` to **<flags>**. This value would get passed to the host's `updconfig` file. This is not used on *fnkits* (see section 23.1.5 *Adding Products to fnkits.fnal.gov*).

- p "<description>"** Specifies product description to set in declaration on distribution node. Not fully implemented; currently just passed through to **UPS**.

- P** Prevents **UPS** from searching in a local database, and thus requires **UPS** to rely solely on information supplied on the command line to locate the product instance to upload. Even if selected instance happens to be declared locally, all database information is ignored.
- q <qualifierList>** Finds product instance on local node with the specified qualifiers (required and/or optional), and sets instance's qualifiers on distribution node.
- r <prodRootDir>** Specifies the local product root directory (generally used with unwound products that are not declared locally; if product is declared locally, can be used to override the product root directory set in the local declaration)
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to "test" on local node, and chains it to "test" on the distribution node. If **-P** used, ignores any local chain, chains selected instance to "test" on the distribution node.
- T <archFilePath>** Specifies location of archive file on local node
- U <upsDir>** Specifies location of `ups` directory on local node; default value is `ups`, relative to the product root directory (generally used with products that are not declared locally; if product instance is declared locally, can be used to override the `ups` directory path set in the local declaration)
- v(vvv)** Prints out extra debugging information.
- V** Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen
- z <databaseList>** Specifies the local database(s) in which to look for the product instance to upload

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavorList>** Finds product instance of specified flavor on local node, and sets instance's flavor on distribution node. If multiple values specified, `upd addproduct` is run once for each flavor.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families. If multiple values specified, `upd addproduct` is run once for each value.

Can be used alone (without an accompanying number option) if product instance is locally declared and **-P** is not used. In this case, the best match instance of the flavor family on the local node is picked, and it gets declared on distribution node as that same "best match" flavor.

- If used with any of **-0**, **-1**, **-2**, **-3**, product instance gets declared on distribution node as specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**. If multiple values specified, **upd addproduct** is run once for each flavor, according to the accompanying number option.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
 - 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
 - 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
 - 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

23.1.4 More Detailed Description

About the tar file

It is optional to create a tar file of your product prior to running **upd addproduct**. **upd addproduct** will create one for you if it knows the location of the product root directory. It can find this information in two ways:

- If the product instance has been declared to a local **UPS** database listed in **\$PRODUCTS** ahead of time, and the product root directory (**PROD_DIR**) appears in the declaration, **UPD** can pick it up. You can check this using **ups list -1**.
- You can supply the product root directory on the **upd addproduct** command line using the **-r** option.

When it is left to **UPD** to create the tar file, it makes the tar file on the local node in the local **\$TMPDIR** area. In the current release of **UPD**, you cannot choose which files to include in a tar file made this way; all files get included except **CVS** directories and core files. The old-style **upd_files.dat** is obsolete.

upd addproduct unwinds the **ups** directory on the distribution node, if it was included in the tar file, thereby making the directory and its contents available for individual file retrieval via **upd fetch** (see section 23.6 *upd fetch*).

About Chains

Chain information remains identical for the added product instance on the local and distribution nodes under most circumstances. If **-P** is used, local chain information is ignored, but can be set on the distribution node. You can use **upd modproduct** afterwards to change the chain (see section 23.10 *upd modproduct*).

Internal Processes

The `upd addproduct` command operates by making a series of network connections to the server. All calls are made from the client system to the distribution server, who reports back results on the same data channel:

- The Web server on the distribution node is called, and a script called `ups.cgi` is used to determine if the specified product instance already exists on the distribution node. If it exists, **UPD** on the client machine prints an error and exits. If it doesn't exist, the process continues.
- The anonymous **FTP** server on the distribution node is called, and the product tar file (if any) is transferred from the user node into `/incoming`.
- The Web server is called, and `upd.cgi` is used to call `upd move_archive_file`. This script makes a product directory for the instance on the distribution node (as defined by the distribution node's `updconfig` file), installs the tar file as `${UPS_PROD_DIR}.tar` (or `${UPS_PROD_DIR}.tar.gz` or `${UPS_PROD_DIR}.zip`, according to its suffix), and unwinds part of the tar file (to make the `README` file and the `ups` and `man` directories available, if present).
- The script `upd.cgi` reports back the database, product directory, and tar file location to the client `upd addproduct` command.
- The anonymous **FTP** server is called, and the product `ups` directory tar file is uploaded to `/incoming`. (If the user specified a `ups` directory, it gets uploaded over the one that was unwound from the tar file.)
- The Web server is called, and `upd.cgi` is used to call `upd moved_ups_dir`. This script makes a `ups` directory on the distribution node for the product (as defined by the `updconfig` file) and unwinds the `ups` directory tar file.
- The script `upd.cgi` reports back the database and `ups` directory to the client `upd addproduct` command.
- The **FTP** server and Web server are similarly called to install the table file.
- Finally, the Web server is called and `ups-decl.cgi` is used to declare the product into the distribution database.

A subset of these steps is performed to execute `upd modproduct` or to add a product that has a subset of these elements (e.g., one that does not include a `ups` directory).

23.1.5 Adding Products to `fnkits.fnal.gov`

The central Fermilab Computing Division product distribution server, `fnkits.fnal.gov`, recognizes several different categories of product:

default	regular products added to the <code>KITS</code> database for distribution to any on-site or registered off-site node ¹ .
FermiTools	locally-developed and supported software packages that we make available to the public
proprietary	products for which Fermilab has a limited number of licenses

1. See the **Product Distribution Platform Registration Request** form at http://www.fnal.gov/cd/forms/upd_registration.html.

fnalonly	products accessible only to the <code>fnal.gov</code> domain
usonly	US-only (United States only) products are accessible only to U.S. government (<code>.gov</code>) and military (<code>.mil</code>) domains

Most products fall into the default category, and can be added normally. For the other categories, you must first fill out the **Special UPD Product Registration** form (at <http://fnkits.fnal.gov/specialprod.html>) indicating which category of product it is, and submit the form. Then when you receive an email message saying that your product has been registered as a special product, go ahead and add it to *fnkits*. Do not use any special options (i.e., `-O "options"`) with `upd addproduct`; your product will automatically be configured to handle the special requirements according to your selection on the form.

23.1.6 upd addproduct Examples

Add locally-declared product using defaults

```
% upd addproduct foo v1_0 -2
```

UPD looks in `$PRODUCTS` to find the product `foo v1_0` for the `-2` flavor level of the local machine. If necessary, it makes a tar file, adds it to the default distribution node *fnkits*, and declares it there. This will work if:

- the local database in which `foo` is declared is listed in `$PRODUCTS` (necessary for tar file creation)
- `foo` has its `ups` directory (in addition to all the product files) under the product root directory, and
- its table file is in one of the default locations (under either the `ups` directory or `$PRODUCTS/foo`).

If the command succeeds, **UPD** returns a message indicating that the product was successfully transferred and declared.

Add locally-declared, unwound product for several flavors, using defaults

```
% upd addproduct foo v1_0 -f IRIX:SunOS:OSF1
```

This example is similar to the first, but shows declaring the product on *fnkits* for three different flavors at the `-1` level. `upd addproduct` gets run three times, once for each flavor.

Add undeclared, unwound product

```
% upd addproduct foo v1_0 -P -2 -m foo.table -M ups \
-r /path/to/foo/prodrootdir -U /path/to/foo/prodrootdir/ups
```

This time the product has not been declared to a local database. Therefore **UPS/UPD** cannot determine where to find the product root directory (`-r`), the table file (`-m` and `-M`) or the `ups` directory (`-U`) on the local node. Again, the flavor is the `-2` flavor level of the local machine. We include `-P` to ensure that no instance declared in the database(s) can be selected in place of the one specified.

Add locally declared tar file

```
% upd addproduct foo v1_0 -2 -T /tmp/foo_v1_0_SunOS+5.tar
```

For this example, we assume the product instance was declared to a local **UPS** database before the tar file was created. The tar file includes the entire structure under the product root directory. **UPD** picks up the pre-made tar file from the local machine in `/tmp/foo_v1_0_SunOS+5.tar` (specified using `-T`), adds it to *fnkits* (no `-h`), and declares it with the `-2` flavor level of the local machine, no chain, and no qualifiers.

Add undeclared product with external ups directory but no table file

```
% upd addproduct footwo v1_0 -P0 -h dist_node.fnal.gov \  
-T /tmp/footwo_v1_0_NULL.tar -U /local/path/to/ups/dir
```

UPD relies solely on information supplied on the command line to execute this command (`-P`). It picks up the tar file (path given via `-T`) of product **footwo** v1_0, flavor NULL (`-0`). No table file is specified (no `-m` or `-M`), therefore **UPD** doesn't look for one. This product has a `ups` directory external to the product root directory (given via `-U`). **UPD** adds the product to the (fictional) node *dist_node.fnal.gov* (given via `-h`).

Add undeclared product consisting only of a table file

```
% upd addproduct foothree v1_0 -Pf IRIX -m foothree.table \  
-M /local/path/to/table/file
```

UPD relies solely on information supplied on the command line to execute this command (`-P`). It picks up the product, **foothree** v1_0 of flavor IRIX, which consists only of a table file (it may be a bundled product). The `-m` and `-M` options are included to specify the table file name and location, there is no product root directory, and thus no `-r`. **UPD** adds it to *fnkits* and declares it with the flavor IRIX, no chain and no qualifiers.

The system returns a notice message saying there is no product root directory. This is correct behavior, and is expected.

23.2 upd cloneproduct

The `upd cloneproduct` command creates a new product instance (the target instance) on a distribution node by copying one that is already there (the source instance) and changing one or more of its identifying elements.

23.2.1 Command Syntax

```
% upd cloneproduct <flavor_option> [<source_options>] <product> \  
    [<version>] -G "<target_options>"
```

23.2.2 All Valid Options

- `-?` ("`-?`" for `cs`) Prints command description and option usage information to screen
- `-f <flavorList>` Described below under "The flavor options".
- `-G "<options>"` Specifies options to be passed to the `ups declare` command on the distribution node for the target instance; see below
- `-h <host>` Specifies product distribution host to which you are adding the product; the default is `fnkits.fnal.gov`. Usually just the plain host name is required, however the `-h` option can always specify any of the following:
 - a plain host name; (e.g., `fred.sub.domain`, or just `fred` if `sub.domain` is `fnal.gov`)
 - a host and Webserver port number (e.g., `fred.sub.domain:8080`)
 - a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)
- `-i` Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one, operation continues to the next.
- `-q <qualifierList>` Finds source product instance on distribution node with the specified qualifiers (required and/or optional)
- `-v(vvv)` Prints out extra debugging information

The flavor options

Flavor may be specified using `-f`, or using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with `-f` or with each other.

- `-f <flavorList>` Specifies flavor of source product on the distribution node. If multiple values specified, `upd cloneproduct` is run once for each flavor.
- `-0` Specifies flavor of source product on the distribution node as `NULL`; equivalent to `-f NULL`

- 1 Specifies flavor of source product on the distribution node as the OS value of local machine up to the generic OS (e.g., equivalent to **-f SunOS**)
- 2 Specifies flavor of source product on the distribution node as OS value of local machine up to the major release (e.g., equivalent to **-f SunOS+5**)
- 3 Specifies flavor of source product on the distribution node as most significant OS specification of local machine or its full specification (e.g., equivalent to **-f SunOS+5.6**)

23.2.3 Options Valid with -G

In order to distinguish the target product instance from the source, the declarations for the two instances must differ by at least one instance-identifying element. The **-G** option provides the means to specify the target instance identifiers; it takes a list of **ups declare** command line elements as an argument. Any identifier not specified via **-G** retains the value of the source instance. The elements valid for use with **-G** include **<product>**, **<version>** and the following subset of the **ups declare** options:

```
-A <nodeList>, -c, -d, -D <origin>, -f <flavor>, -g <chainName>, -n, -o, -O "<flagList>", -p "<description>", -q <qualifierList>, -t, -z <databaseList>, -0, -1, -2, -3
```

See section 22.5 *ups declare* for details on each option. If the argument to **-G** includes the product version, the product name must be included ahead of the version; the first unflagged element is always interpreted as the product name and the second as the version.

23.2.4 upd cloneproduct Example

```
% upd cloneproduct -f NULL jfc v1_0 -G "--f CYGWIN32_NT"
```

In this example, **UPD** finds the (OS-independent) product **jfc** version **v1_0** of flavor **NULL**, and clones a new instance with all identifiers the same except for the flavor. The new instance has the flavor **CYGWIN32_NT**, for NT users. Going to **CYGWIN32_NT** presents a problem, so we provide some further explanation:

This product contains **java** classes. For **java** to dynamically load class libraries, it looks in an environment variable called **CLASSPATH** which contains the directories with the **java** classes you want to use. On UNIX, the directories in **CLASSPATH** need to be colon (**:**) separated, but in **CYGWIN**, they need to be semi-colon (**;**) separated.

We've handled this by setting a delimiter variable in the product's table file. One instance in the table file is defined for **NULL** (using colon delimiters), and one is for **CYGWIN32_NT** (using semi-colons). For example, in the table file under the **SETUP** action for **Flavor=NULL** we have:

```
envPrepend (CLASSPATH, ${UPS_PROD_DIR}/swingall.jar, ":")
```

and under **Flavor=CYGWIN32_NT**, it is changed to:

```
envPrepend (CLASSPATH, $jfc_cpath, ";")
```

Everything in the two product instances is exactly the same, except the delimiters.

23.3 upd delproduct

The `upd delproduct` command deletes a product declaration from a distribution database. It also removes any associated tar file, table file and/or `ups` directory. The product subdirectory itself does not get deleted.

23.3.1 Command Syntax

```
% upd delproduct -f <flavor_option> [<other_options>] <product> \  
  <version>
```

23.3.2 Commonly Used Options

See section 23.3.3 *All Valid Options* for descriptions of each option.

```
-f <flavorList>    Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                  -2, -3)  
  
-h <host>  
  
-q <qualifierList>
```

23.3.3 All Valid Options

<code>-?</code> (" <code>-?</code> " for <code>csh</code>)	Prints command description and option usage information to screen
<code>-f <flavorList></code>	Described below under "The flavor options".
<code>-h <host></code>	Specifies product distribution host from which you are deleting the product; the default is <i>fnkits.fnal.gov</i> . Usually just the plain host name is required, however the <code>-h</code> option can always specify any of the following: <ul style="list-style-type: none">- a plain host name; (e.g., <i>fred.sub.domain</i>, or just <i>fred</i> if <i>sub.domain</i> is <i>fnal.gov</i>)- a host and Webserver port number (e.g., <i>fred.sub.domain:8080</i>)- a full URL to the <code>ups.cgi</code> cgi script (e.g., <code>http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi</code>)
<code>-H <flavorList></code>	Described below under "The flavor options".
<code>-i</code>	Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one, operation continues to the next.
<code>-O "<flags>"</code>	Sets the value of <code>\$UPS_OPTIONS</code> to <code><flags></code> . This value would get passed to the host's <code>updconfig</code> file. This is not used on <i>fnkits</i> (see section 23.1.5 <i>Adding Products to fnkits.fnal.gov</i>).

- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional)
- s** Lists what command would do; but does not execute the command
- v(vvv)** Prints out extra debugging information.

The flavor options

Flavor may be specified using **-f**, using **-H** in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavorList>** Finds product instance of specified flavor on distribution node. If multiple values specified, **upd delproduct** is run once for each flavor.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families.

Must be used with one of **-0**, **-1**, **-2**, **-3**, to specify flavor absolutely. Used this way, it finds product instance of specified flavor on distribution node; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**. If multiple values specified, **upd delproduct** is run once for each flavor, according to the accompanying number option.
- 0** Finds product instance of NULL flavor on distribution node; equivalent to **-f NULL**
- 1** Finds product instance (on distribution node) of local OS flavor, specified up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Finds product instance (on distribution node) of local OS flavor, specified up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3** Finds product instance (on distribution node) of local OS flavor, specified up to the most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

23.3.4 upd delproduct Example

```
% upd delproduct foo v1_0 -cf SunOS+5
```

This command deletes the product **foo v1_0** declared as “current” for the flavor SunOS+5. (Since the version is specified, the **-c** is unnecessary, but harmless.) The command syntax is the same whether the product is in archived format, is unwound or consists of just a table file.

23.4 upd depend

The **upd depend** command executes **ups depend** on a product distribution database. The **ups depend** command lists product dependencies of the specified product instance(s) as declared locally. See section 22.6 *ups depend* for more information and examples.

upd install runs this command internally to determine what dependencies to install with the requested product. Product installers can use it to see what products **upd install** would distribute.

23.4.1 Command Syntax

```
% upd depend [-h <host>] [<ups_depend_options>] <product> \  
  [<version>]
```

23.4.2 Options

The **upd depend** command uses all the same options as **ups depend**, plus **-h** (described below). See section 22.6 *ups depend* for the list of options and their descriptions. The difference in the options' functionality is that they apply to the distribution node rather than to the local node.

-h <host> Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:

- a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
- a host and Webserver port number (e.g., *fred.sub.domain:8080*)
- a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)

23.4.3 upd depend Examples

```
% upd depend exmh
```

This example lists all the dependencies declared for the default instance of **exmh** on the default host *fnkits*:

```
exmh v2_0_2 -f NULL -z /ftp/upsdb -g current  
|__expect v5_25 -f SunOS+5 -z /ftp/upsdb -g current  
| |__tk v8_0_2 -f SunOS+5 -z /ftp/upsdb  
| |__tcl v8_0_2 -f SunOS+5 -z /ftp/upsdb  
|__mh v6_8_3c -f SunOS+5 -z /ftp/upsdb -g current  
| |__mailtools v2_3 -f NULL -z /ftp/upsdb -g current  
|__mimetools v2_7a -f SunOS+5 -z /ftp/upsdb -g current  
|__glimpse v3_0a -f SunOS+5 -z /ftp/upsdb -g current  
|__www v3_0 -f NULL -z /ftp/upsdb -g current  
| |__lynx v2_8_1 -f SunOS+5 -z /ftp/upsdb -g current  
|__ispell v3_1b -f SunOS+5 -z /ftp/upsdb -g current
```

To specify a different host, use the **-h** option, e.g.,

```
% upd depend -h dist_node exmh
```

If a chain rather than a version number is used to specify the instance (as is the case for the default “current” instance), then the chain appears in the output line for the product (notice the **-g current** in the first line); otherwise the chain is not listed.

We refer you to section 22.6 *ups depend* for more examples, as the two commands use the same syntax and options (except for **-h**) and produce similar output.

23.5 upd exist

The **upd exist** command runs **ups exist** on a product distribution node. The **ups exist** command is used to test whether a **setup** command issued on the local machine with the same command line elements is likely to succeed. See section 22.7 *ups exist* for more information and examples.

upd exist can be used to test for the existence of a particular product instance on a distribution node, prior to installing it. It can be used whether the distribution database is “live” or in archive format.

23.5.1 Command Syntax

```
% upd exist [-h <host>] [<ups_exist_options>] <product> \  
  [<version>]
```

23.5.2 Options

The **upd exist** command uses all the same options as **ups exist**, plus **-h** (described below). See section 22.7 *ups exist* for the list of options and their descriptions. The difference in the options’ functionality is that they apply to the distribution node rather than to the local node.

-h <host> Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:

- a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
- a host and Webserver port number (e.g., *fred.sub.domain:8080*)
- a full URL to the *ups.cgi* cgi script (e.g., *http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi*)

23.5.3 upd exist Examples

This command is rarely used from the command line, and is more useful in scripts where a failed setup could cause the script to abort. When issued from the command line, it returns no output if the command succeeds.

In the C shell family **upd exist** sets the `$status` variable to `0` if it was able to create the temporary file, or to `1` for error. In the Bourne shell family, it sets the `$?` variable similarly. As an example, we can run **upd list** and find that there is a current instance of the product **tex** for the flavor IRIX+6 but not for IRIX+6.2. Running **upd exist** for each flavor, we see that the variables get set accordingly:

For the C shell family:

```
% upd exist tex -f IRIX+6; echo $status
```

0

```
% upd exist tex -f IRIX+6.2; echo $status
```

1

For the Bourne shell family:

```
$ upd exist tex -f IRIX+6; echo $?
```

0

```
$ upd exist tex -f IRIX+6.2; echo $?
```

1

23.6 upd fetch

The **upd fetch** command performs either of the following functions:

- If **-J** is used, it retrieves a single file or directory from a product distribution database, and downloads it to the user node, placing it relative to the current working directory.
- If **-J** is not used, returns a recursive list of directories and files that are available for retrieval from the product distribution node. Warning: When using a live distribution database, this list may be very long.

The **upd fetch** command cannot retrieve files from within a tar file. In an archive distribution database, typically the only files provided in the appropriate format are the `README` and `INSTALL_NOTE` files, the `ups` subdirectory files, and the table and version files.

This command is used internally by **UPD**, and only rarely run manually.

23.6.1 Command Syntax

```
% upd fetch [<options>] [-J fileName] <product> [<version>]
```

23.6.2 Commonly Used Options

See section 23.6.3 *All Valid Options* for descriptions of each option.

```
-f <flavorList>    Or one of -0, -1, -2, -3, or -H (alone or together with one of  
                  -0, -1, -2, -3)  
-g <chainName>    Or one of -c, -d, -n, -o, -t  
-h <host>  
-J <fileName>  
-q <qualifierList>
```

23.6.3 All Valid Options

```
-? ("-" for csh)  Prints command description and option usage information to screen  
-c                Finds product instance chained to "current" on the distribution node  
-d                Finds product instance chained to "development" on the distribution  
                  node  
-f <flavorList>  Described below under "The flavor options".  
-g <chainName>   Finds product instance chained to <chainName> on the distribution  
                  node; can be a list of chain names
```

- h <host>** Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:
 - a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
 - a host and Webserver port number (e.g., *fred.sub.domain:8080*)
 - a full URL to the `ups.cgi` script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)
- H <flavorList>** Described below under “The flavor options”.
- i** Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one, operation continues to the next.
- J <fileName>** Specifies an individual file to fetch from the distribution node (it does not accept a list of files, however you can retrieve a file with a colon in its name). The argument `@table_file` is valid for specifying the table file.
- n** Finds product instance chained to “new” on the distribution node
- o** Finds product instance chained to “old” on the distribution node
- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional)
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test” on the distribution node
- v(vvv)** Prints out extra debugging information.

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).

- f <flavorList>** Finds product instance of specified flavor on distribution node. If multiple values specified, `upd fetch` is run once for each flavor.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families.

Can be used alone (without an accompanying number option). In this case, the best match instance of the flavor family on the distribution node is picked.

Can also be used with one of **-0**, **-1**, **-2**, **-3**, to specify flavor absolutely. Used this way, it finds product instance of specified flavor on distribution node; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**. If multiple values specified, `upd fetch` is run once for each flavor, according to the accompanying number option.

- 0 Finds product instance of NULL flavor on distribution node; equivalent to **-f NULL**
- 1 Finds product instance (on distribution node) of local OS flavor, specified up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2 Finds product instance (on distribution node) of local OS flavor, specified up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3 Finds product instance (on distribution node) of local OS flavor, specified up to the most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

23.6.4 upd fetch Examples

Output a list of files and directories

```
% upd fetch teledata
```

This first example illustrates the command output when the **-J** option is omitted on a request to a live distribution database. Nothing actually is retrieved when **-J** is absent. The output is a recursive list of directories and files that are available for individual retrieval (list edited for brevity):

```
Listing of table_dir [/ftp/products/teledata/v1_0/NULL]:
total 26442
drwxrwx---  3 updadmin upd          512 May 12 1999 teledata_v1_0_NULL
-rw-rw-r--  1 updadmin upd          712 May 12 1999 teledata_v1_0_NULL.table
-rw-rw----  1 updadmin upd          712 Jan 28 1999 teledata_v1_0_NULL.table.old
-rw-rw----  1 updadmin upd    13516800 May 12 1999 teledata_v1_0_NULL.tar
-rw-rw----  1 updadmin upd          9728 May 12 1999 teledata_v1_0_NULL.ups.tar

teledata_v1_0_NULL:
total 12
-rw-r--r--  1 updadmin upd          4513 Feb  9 1999 README
drwxr-xr-x  2 updadmin upd          512 Jan 28 1999 ups

teledata_v1_0_NULL/ups:
total 12
lrwxrwxrwx  1 updadmin upd           9 May 12 1999 INSTALL_NOTE -> ../README
-rwxr-xr-x  1 updadmin upd          541 May 12 1999 configure
-rwxr-xr-x  1 updadmin upd          568 May 12 1999 current
-rw-r--r--  1 updadmin upd          784 Jan  5 1999 teledata.table
-rwxr-xr-x  1 updadmin upd          209 May 12 1999 unconfigure
-rwxr-xr-x  1 updadmin upd          212 May 12 1999 uncurrent

Listing of @ups_dir [/ftp/products/teledata/v1_0/NULL/teledata_v1_0_NULL/ups]:
total 12
...
```

```
Listing of @prod_dir [/ftp/products/teledata/v1_0/NULL/teledata_v1_0_NULL]:
total 12
-rw-r--r--  1 updadmin upd          4513 Feb  9  1999 README
drwxr-xr-x  2 updadmin upd          512 Jan 28  1999 ups

ups:
total 12
...
```

Retrieve a file

```
% upd fetch -f IRIX+6 -J README tex v3_14159
```

This example shows the retrieval of a `README` file (`-J README`) for the product `tex`. We specifically request the `README` pertaining to the flavor `IRIX+6` for the product version `v3_14159`. The file will be copied to the current working directory. The system provides some informational output:

```
informational: transferred README
from fnkits.fnal.gov:/ftp/products/tex/v3_14159/IRIX+6/tex_v3_14159_IRIX+6
to ./README
```

Retrieve the table files for several flavors of a product

```
% upd fetch -H SunOS+5:IRIX+6:Linux+2:OSF1+V4 -J @table_file \
tex v3_14159
```

This example retrieves the table file(s) for the best match product instances of `tex v3_14159` for the listed flavor families. Depending on how the product was configured, the same table file may be used for all, or they may be separate files. The file(s) will be copied to the current working directory.

23.7 upd get

The `upd get` command runs `ups get` on a product distribution node. Currently they can only be used with the `-F` option. `upd get -F` lists any files on the distribution node which are to be distributed with the specified product instance(s) and which are maintained outside of the product root directory. The list does not include table files, for which the location is maintained in the version file. See section 22.9 *ups get* for more information and examples.

The `ups get` and `upd get` commands were designed primarily for use by **UPD**, which calls it internally. As such they are rarely used outside of that context. In a future release, `ups get` may acquire additional functions.

23.7.1 Command Syntax

```
% upd get -F [-h <host>] [<ups_get_options>] <product> [<version>]
```

23.7.2 Options

The `upd get` command uses all the same options as `ups get`, plus `-h` (described below). See section 22.9 *ups get* for the list of options and their descriptions. The difference in the options' functionality is that they apply to the distribution node rather than to the local node.

-h <host> Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the `-h` option can always specify any of the following:

- a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
- a host and Webserver port number (e.g., *fred.sub.domain:8080*)
- a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)

23.8 upd install

The `upd install` command retrieves a product instance and by default its dependencies, as needed, from a product distribution node. It installs the retrieved instances on the local node, declares them to a local **UPS** database, and optionally runs commands needed to resolve dependencies.

23.8.1 Command Syntax

```
% upd install [<options>] <product> [<version>]
```

23.8.2 Commonly Used Options

See section 23.8.3 *All Valid Options* for descriptions of each option.

```
-f <flavorList>    Or one of -0, -1, -2, -3, or -H (alone or together with one of  
                  -0, -1, -2, -3)  
-g <chainName>    Or one of -c, -d, -n, -o, -t  
-G "<options>"  
-h <host>  
-i  
-I  
-j  
-q <qualifierList>  
-R  
-X  
-z <databaseList>
```

23.8.3 All Valid Options

-? ("-" for <code>cs</code>)	Prints command description and option usage information to screen
-c	Finds product instance chained to "current" on the distribution node
-C	Prevents execution of the CONFIGURE action during the initial local product declaration run by <code>upd install</code> . It also prevents execution of any action that would otherwise be called by options listed in the <code>-G</code> argument (e.g., <code>-G "-c"</code> normally would call CURRENT, but in this case acts like <code>-G "-c -C"</code> , which would not call CURRENT).
-d	Finds product instance chained to "development" on the distribution node
-f <flavorList>	Described below under "The flavor options".

- g <chainName>** Finds product instance chained to **<chainName>** on the distribution node; can be a list of chain names
- G "<options>"** Specifies options to be passed to the local **ups declare** command; see below
- h <host>** Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:
 - a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
 - a host and Webserver port number (e.g., *fred.sub.domain:8080*)
 - a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)
- H <flavorList>** Described below under "The flavor options".
- i** Ignores errors; if a dependency can't be downloaded or installed locally, operation continues.
- I** Unwinds product tar file on the local node after transmission (default is to unwind during transmission). `$TMPDIR` is used as the tar file destination. Use this if your network is unreliable. Note that you need twice the disk space for installs.
- j** Ignores dependencies, installs specified product only
- m <tableFileName>** Specifies table file name for the product instance on the distribution node
- M <tableFileDir>** Specifies table file directory for the product instance on the distribution node
- n** Finds product instance chained to "new" on the distribution node
- o** Finds product instance chained to "old" on the distribution node
- O "<flags>"** Sets the value of `$UPS_OPTIONS` to **<flags>**. You'd use this if your `updconfig` file had corresponding stanzas.
- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional)
- r <prodRootDir>** Specifies the product root directory for the product instance on the distribution node
- R** Downloads and installs only the required dependencies of the specified product.
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to "test" on the distribution node
- U <upsDir>** Specifies location of `ups` directory; default value is **ups**
- v(vvv)** Prints out extra debugging information.

-X Executes the generated **ups declare** commands needed to resolve dependencies. If **-X** is not included, **ups install** echoes the commands to the screen.

-z <databaseList> Specifies the local database(s) in which product and its dependencies can be installed. See section 26.1 *Database Selection Algorithm* for info on how the database is chosen.

If **-z** is specified as a normal option to **upd install**, then it determines the database in which to look for the **UPD** configuration; if not, then **\$PRODUCTS** is used for this purpose. If specified within **-G** construction, then it gets handed to **ups declare** and is used for the second declaration. It may be specified in both places, but if not done with appropriate forethought, this can lead to errors.

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).



-f <flavorList> Finds product instance of specified flavor on distribution node, and sets flavor by default on local node. If multiple values specified, **upd install** is run once for each flavor. Avoid using **-f** except in a **-G** argument string, because it can cause flavor mismatches between main product and dependencies.

-H <flavorList> Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families.

Can be used alone (without an accompanying number option). In this case, the best match instance of the flavor family on the distribution node is picked.

Can also be used with one of **-0**, **-1**, **-2**, **-3**, to specify flavor absolutely. Used this way, it finds product instance of specified flavor on distribution node; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**. If multiple values specified, **upd install** is run once for each flavor, according to the accompanying number option.

-0 Finds product instance of NULL flavor on distribution node; equivalent to **-f NULL**

-1 Finds product instance (on distribution node) of local OS flavor, specified up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.

-2 Finds product instance (on distribution node) of local OS flavor, specified up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.

-3

Finds product instance (on distribution node) of local OS flavor, specified up to the most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as `IRIX+6.2`.

23.8.4 Options Valid with -G

The `-G` option is provided to allow you to pass **UPS** options to the `ups declare` command. The `upd install` command first declares the product without referencing `-G`, then does a second declaration with this information. Any identifier not specified via `-G` retains its previous value. The elements valid for use with `-G` include `<product>`, `<version>` and the following subset of the `ups declare` options:

`-A <nodeList>`, `-c`, `-d`, `-D <origin>`, `-f <flavor>`, `-g <chainName>`, `-n`, `-o`, `-O "<flagList>"`, `-p "<description>"`, `-q <qualifierList>`, `-t`, `-z <databaseList>`, `-0`, `-1`, `-2`, `-3`

If you include no instance identifiers within `-G` (i.e., exclude flavor, `-q`, and `-z`) so that the second declaration is just modifying the previously declared instance (e.g., setting a chain), then the options `-A`, `-D` and `-p` are ignored.

See section 22.5 *ups declare* for details on each option. If the argument to the `-G` option includes the product version, it must also include the product name ahead of the version; the first unflagged element is always interpreted as the product name and the second as the version.

Note that options included here apply to the product's dependencies as well as to the main product. Make sure that you don't assign chains or flavors to dependencies required by other products in such a way as to make those products break.

23.8.5 More Detailed Description

The `upd install` command performs the following series of functions:

- retrieves the specified product instance, and by default its dependencies, from a distribution node
- installs the product on the user node according to the node's **UPD** configuration
- unwinds the product if transferred in tar format
- declares the product to a local database (by calling `ups declare`); the database specified in the node's **UPD** configuration is the default, but it can be overridden on the command line
- prints to screen the commands you will need to issue in order to resolve dependencies (Usually these are chain declarations. Most of the time dependencies are based on chains not versions.)

Internal Processes

The `upd install` command operates by making a series of network connections to the server. All calls are made from the client system to the distribution server, who reports back results on the same data channel:

- The Web server on the distribution node is called, and `ups.cgi` is used to determine if the product instance in question exists on the server, what its dependencies are. A call to the local **UPS** determines whether the product and its dependencies exist on the user node. For the product itself and for each dependency not found on the user node, the remaining steps are taken:
- The Web server is called, and `ups.cgi` is used to determine particular details of the product on the distribution node (e.g., archive file location, product root directory, `ups` directory, and so on). If no archive file location is given, **UPD** manufactures a tar file that should work, assuming the **FTP** server can make a tar file of directories on the fly.¹ The tar file gets named according to the convention: `ftp://host/$UPS_PROD_DIR/..tar` (a “.” for the path, followed by “.tar”).
- The **FTP** server is used to transfer and unwind the archive file, an archive of the `ups` directory, and the table file for the product.
- **UPD** declares the product on the local system.

23.8.6 upd install Examples

Default installation

```
% upd install tex
```

In this example, **UPD** downloads the current instance of the product `tex` (for the flavor of the local machine) from the default distribution node `fnkits`, and installs and declares it on the local machine using the defaults in place. Dependencies, as needed, also get installed.

Install only default instance into specified database and make “current”

```
% upd install -z $MYDB -G "-c" -j tex
```

```
% upd install -z /path/to/my/database -G "-g current" -j tex
```

These two examples are equivalent assuming `$MYDB` is set to the path shown. They install only the product `tex` (`-j` specifies no dependencies) and include the `-z` option to specify the target database. The product `tex` is constrained to exist in the specified database, and it gets chained to “current”.

Install multiple flavors of a product

```
% upd install perl v5_005 -H CYGWIN32_NT:Linux+2:SunOS+5:IRIX+6 -C
```

UPD retrieves the instances of `perl` version `v5_005` for the listed flavors, and all dependencies (correctly matched for flavor), as needed. If `-f` were used in place of `-H` here, all the dependencies installed would be of the best match flavor to the local machine, and most or all product instances wouldn't work right.

1. A **WU-FTP** compatible **FTP** server is used to make tar files “on the fly”.

The **-C** option prevents execution of the CONFIGURE action, which may be OS-specific. In this case, the installer needs to login to one machine of each flavor in the cluster and manually run **ups configure** for **perl. ups configure** is described in section 22.3 *ups configure*.

Install product plus required dependencies

```
% upd install exmh v2_0_2 -h dist_node -R
```

This command installs the product **exmh** version **v2_0_2** for the best match flavor of the local machine. It uses the distribution node *dist_node*. The option **-R** causes only the product and its required dependencies (as needed) to get installed.

Install a product from a CD-ROM

A distribution database can be on CD-ROM. See <http://www.fnal.gov/docs/products/ups/ReferenceManual/misc/cdrom.html> for more information. To install a product from CD-ROM, first insert the CD-ROM and run the **mount** command:

```
% mount -t iso9660 /dev/cdrom /path/to/db/on/cdrom
```

Then to install a product from the CD-ROM, run the **upd install** command normally, but include the **-h** option pointing to the database on the CD-ROM as shown:

```
% upd install -h file://localhost/path/to/db/on/cdrom ...
```

The Linux CD-ROM images created by the Computing Division mount the distribution database on the CD-ROM to `/tmp/rhimage/products/db`. So, for example, the commands would look like:

```
% mount -t iso9660 /dev/cdrom /tmp/rhimage
```

```
% upd install -h file://localhost/tmp/rhimage/products/db  
<product> <version> [-G <options>] ...
```

23.9 upd list

The `upd list` command performs `ups list` on a distribution database.

The `ups list` command returns information about the declared product instances in a local UPS database. Two output styles are provided: a formatted one that is easy for users to read, and a condensed one for parsing by a subsequent command or a script.

See 22.11 *ups list* for more information and examples.

23.9.1 Command Syntax

```
% upd list [-h <host>] [<ups_list_options>] [<product>] \  
  [<version>]
```

23.9.2 Options

The `upd list` command uses all the same options as `ups list`, plus `-h` (described below). See section 22.11 *ups list* for the list of options and their descriptions. The difference in the options' functionality is that they apply to the distribution node rather than to the local node.

<code>-h <host></code>	Specifies product distribution host; the default is <i>fnkits.fnal.gov</i> . Usually just the plain host name is required, however the <code>-h</code> option can always specify any of the following: <ul style="list-style-type: none">- a plain host name; (e.g., <i>fred.sub.domain</i>, or just <i>fred</i> if <i>sub.domain</i> is <i>fnal.gov</i>)- a host and Webserver port number (e.g., <i>fred.sub.domain:8080</i>)- a full URL to the <code>ups.cgi</code> cgi script (e.g., <code>http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi</code>)
------------------------------	--

23.9.3 upd list Examples

We refer you to section 22.11 *ups list* for many examples. Simply substitute `upd list` for `ups list`, and the commands perform the same function, but on a distribution node. Use the `-h <host>` option to specify a distribution node different from the default *fnkits.fnal.gov*, e.g., the command:

```
% upd list -aK+ -h dist_node emacs
```

This requests the standard output fields for all instances (`-a` for all) of `emacs`, from the distribution node `dist_node`, using the condensed output format (`-K+`):

```
"emacs" "v19_30a" "AIX+3" "" ""  
"emacs" "v19_30a" "IRIX+5" "" ""  
"emacs" "v19_30a" "SunOS+5" "" ""  
...  
"emacs" "v19_34b" "SunOS+5" "" "current"
```


23.10 upd modproduct

The `upd modproduct` command is used to modify a product instance that already exists in a distribution database. It allows you to replace a table file or `ups` directory, or to add or change chain information for the product. To modify any other data, you must delete the product or tar file from the distribution database and add it again with the new values (see sections 23.3 *upd delproduct* and 23.1 *upd addproduct*). `upd modproduct` does not modify product tar files.

23.10.1 Command Syntax

For replacing a table file

```
% upd modproduct <flavor_option> -m <tableFileName> \  
  [-M <tableFileDir>] [<other_options>] <product> <version>
```

Note: You must include the `-m` option specifying the table file name, as there is no default. You must also include `-M` if the table file is not in the current directory.

For replacing a ups directory

```
% upd modproduct <flavor_option> -U <upsDir> [<other_options>]\  
  <product> <version>
```

For adding or changing a chain

```
% upd modproduct <flavor_option> <chain_option> [<other_options>]\  
  <product> <version>
```

23.10.2 Commonly Used Options

See section 23.10.3 *All Valid Options* for descriptions of each option.

For replacing a table file or ups directory

```
-f <flavorList>      Or one of -0, -1, -2, -3, or -H (together with one of -0, -1,  
                    -2, -3)  
-g <chainName>      Or one of -c, -d, -n, -o, -t  
-h <host>  
-m <tableFileName>  
-M <tableFileDir>  
-q <qualifierList>  
-U <upsDir>
```

For adding or changing a chain

- f <flavorList>** Or one of **-0**, **-1**, **-2**, **-3**, or **-H** (together with one of **-0**, **-1**, **-2**, **-3**)
- g <chainName>** Or one of **-c**, **-d**, **-n**, **-o**, **-t**
- h <host>**
- q <qualifierList>**

23.10.3 All Valid Options

- ? ("-" for csh)** Prints command description and option usage information to screen
- c** When adding a chain, assigns the “current” chain to the product instance on the distribution node; when replacing a component, ignored.
- d** When adding a chain, assigns the “development” chain to the product instance on the distribution node; when replacing a component, ignored.
- f <flavorList>** Described below under “The flavor options”.
- g <chainName>** When adding a chain, assigns the **<chainName>** chain to the product instance on the distribution node; when replacing a component, ignored. Can be a list of chain names.
- h <host>** Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:
 - a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
 - a host and Webserver port number (e.g., *fred.sub.domain:8080*)
 - a full URL to the *ups.cgi* cgi script (e.g., *http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi*)
- H <flavorList>** Described below under “The flavor options”.
- i** Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one, operation continues to the next.
- m <tableFileName>** When uploading a table file and/or *ups* directory containing a new table file, specifies local table file name
- M <tableFileDir>** When uploading a table file and/or *ups* directory containing a new table file, specifies local table file directory
- n** When adding a chain, assigns the “new” chain to the product instance on the distribution node; when replacing a component, ignored.
- o** When adding a chain, assigns the “old” chain to the product instance on the distribution node; when replacing a component, ignored.
- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional)

- s** Lists what command would do; but does not execute the command
- t** When adding a chain, assigns the “test” chain to the product instance on the distribution node; when replacing a component, ignored.
- U <upsDir>** When uploading a `ups` directory, specifies its location on local node; default value is `ups` (relative to the product root directory)
- v(vvv)** Prints out extra debugging information.

The flavor options

Flavor may be specified using `-f`, using `-H` by itself or in combination with any of `-0`, `-1`, `-2`, `-3`, or just using one of `-0`, `-1`, `-2`, `-3`. These options are not valid with each other (except `-H` with a number option).

- f <flavorList>** Finds product instance of specified flavor on distribution node. If multiple values specified, `upd modproduct` is run once for each flavor.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families. If multiple values specified, `upd modproduct` is run once for each value.

Must be used with any of `-0`, `-1`, `-2`, `-3`; e.g., `-2H IRIX+6.2` is equivalent to `-f IRIX+6`. If multiple values specified, `upd modproduct` is run once for each flavor, according to the accompanying number option.
- 0** Specifies flavor as NULL; equivalent to `-f NULL`
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to `-f SunOS`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to `-f SunOS+5`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to `-f SunOS+5.6`; if given together with `-H IRIX+6.2`, flavor is then specified as IRIX+6.2.

23.10.4 More Detailed Description

Note that `upd modproduct` cannot query the local **UPS** database to find information the way `upd addproduct` can; all necessary information must be specified on the command line.

A product instance on a distribution node generally has at most one chain associated with it at any time.¹ Whenever you change a chain with `upd modproduct`, you automatically delete any and all previously assigned chain or chains.

When you use `upd modproduct` to replace a `ups` directory:

- If the `ups` directory contains a newer table file that should replace the old one, include the `-m` and `-M` options in the command.
- Do not make a tar file of the `ups` directory on your local machine.

23.10.5 upd modproduct Examples

Replace a table file

```
% upd modproduct foo v1_0 -2 -m v1_0.table -M \  
/local/path/to/foo/ups/dir
```

For this example, we assume that a new table file has replaced the old one in the product instance's local `ups` directory, and it must now be added to the distribution node. In this `upd modproduct` command:

- we identify the product instance associated with the table file: `foo v1_0`, of the level `-2` flavor of the local machine, in `$PRODUCTS`
- `-m v1_0.table` gives the name of the new table file
- `-M /local/path/to/foo/ups/dir` gives the table file directory. Even though the table file is in one of the default locations, the location must be specified unless you issue the command from the directory specified by `-M`. The `upd modproduct` command doesn't query `UPS` for the information.

Replace a ups directory

```
% upd modproduct foofour v1_0 -f SunOS -h dist_node -U \  
/local/path/to/ups/dir
```

In this example, the `upd modproduct` command is used with the `-U` option to replace a product instance's `ups` directory. We illustrate with a product called `foofour v1_0`, flavor `SunOS`, and no qualifiers. We distribute it to `dist_node` using `-h dist_node` (the distribution database is determined by the `UPD` configuration on `dist_node`). It doesn't matter whether the product instance is declared to a `UPS` database listed in `$PRODUCTS`, since `upd modproduct` won't query the database anyway. Regardless of its location, the `ups` directory location must be fully specified.

Declare a chain

```
% upd modproduct foo v1_0 -t -i -f SunOS+5:IRIX+6:Linux+2:OSF1+V4
```

For this example, we assume that the product `foo v1_0` has no chain in its `KITS` declarations (default distribution host used here). In this command we declare a "test" chain for several flavors using the `-t` option (`-g test` would work too). We include `-i` in case any of the flavors doesn't exist; the command will proceed to the next flavor.

1. A product instance can have multiple chains if they are declared together in the same command (e.g., `upd modproduct -g test:current ...`).

Change a declared chain

```
% upd modproduct foo v1_0 -c -f SunOS+5:IRIX+6:Linux+2:OSF1+V4
```

This command changes the “test” chains for **foo** in the above example to “current”. This removes the test chains. Again, **-i** allows the command to proceed in case of errors.

Remove a chain

```
% upd modproduct foo v1_0 -f SunOS+5 -g :
```

This command removes a chain on the specified instance. It doesn’t assign a new one, nor does it assign the existing chain to a different instance. This often generates warnings, but it works and causes no database problems.

23.11 upd reproduct

The `upd reproduct` command is equivalent to a `upd delproduct` followed by a `upd addproduct`. It can be used only when the replacement product instance on the local node has the same set of identifiers as the one on the distribution node destined for removal. It takes the same command line elements as `upd addproduct`. See sections 23.1 *upd addproduct* and 23.3 *upd delproduct* for more information on those commands.

23.11.1 Command Syntax

For replacing with a product that is declared to a local database

An unwound product or a table file:

```
% upd reproduct <flavor_option> [<other_options>] <product> \  
  <version>
```

A product tar file:

```
% upd reproduct <flavor_option> -T <archFilePath> \  
  [<other_options>] <product> <version>
```



Note: One difference from the `upd addproduct` syntax: The flavor option is strictly required here; `upd reproduct` does not default to the current instance on the local system.

For replacing with a product that is not declared to a local database

An unwound product:

```
% upd reproduct [-P] <flavor_option> -r <prodRootDir>    \  
  -m <tableFileName> [-M <tableFileDir>] [<other_options>] \  
  <product> <version>
```

A table file:

```
% upd reproduct [-P] <flavor_option> -m <tableFileName> \  
  [-M <tableFileDir>] [<other_options>] <product> <version>
```

A product tar file:

```
% upd reproduct [-P] <flavor_option> -T <archFilePath>    \  
  -m <tableFileName> [-M <tableFileDir>] [<other_options>] \  
  <product> <version>
```

Notes:

- The `-P` option is not strictly required, however it is recommended. In cases where the local database contains a product instance that also matches the specified options, `-P` ensures that UPD ignores the database and picks up the intended instance.
- If the product includes a table file, you must include the `-m` option specifying its name, as there is no default. You must also include `-M` if the table file is not in the current directory.

23.11.2 Options

See section 23.1.3 *All Valid Options* under 23.1 *upd addproduct*.

23.11.3 upd reproduct Examples

Add and then replace a locally-declared product using defaults

```
% upd addproduct foo v1_0 -2
```

UPD looks in \$PRODUCTS to find the product **foo** v1_0 for the **-2** flavor level of the local machine. If necessary, it makes a tar file, adds it to the default distribution node *fnkits*, and declares it there.

Let's say that after adding the product you discovered it contained the wrong set of executables. After fixing the problem on your local node, you need to replace the product instance on the distribution node. You could just run the command:

```
% upd reproduct foo v1_0 -2
```

23.12 upd update

The **upd update** command retrieves the table file, `ups` directory, or both, for a product instance, and by default for its product dependencies, from a distribution database for the purpose of updating the pre-existing one(s) on a local node.

Overwriting occurs only if the MODIFIED date in the version file that points to the table file on the distribution node is later than that in the corresponding local version file; file timestamps are not used for the comparison. The update will fail if there is no corresponding pre-existing component on the local node.



If you need to update *both* the `ups` directory and the table file for a product, do it at the same time (using the component list **table_file:ups_dir**). Otherwise, the MODIFIED date will not allow the second operation to succeed.

23.12.1 Command Syntax

```
% upd update [<options>] <componentList> <product> [<version>]
```

Components

table_file	the table file
ups_dir	the ups directory

23.12.2 Commonly Used Options

See section 23.12.3 *All Valid Options* for descriptions of each option.

-f <flavorList>	Or one of <code>-0</code> , <code>-1</code> , <code>-2</code> , <code>-3</code> , or <code>-H</code> (alone or together with one of <code>-0</code> , <code>-1</code> , <code>-2</code> , <code>-3</code>)
-g <chainName>	Or one of <code>-c</code> , <code>-d</code> , <code>-n</code> , <code>-o</code> , <code>-t</code>
-h <host>	
-i	
-I	
-j	
-q <qualifierList>	
-R	
-z <databaseList>	

23.12.3 All Valid Options

-? (" <code>-?</code> " for <code>csh</code>)	Prints command description and option usage information to screen
-a	Updates all instances that match the other options given on command line.

- c** Finds product instance chained to “current” on local and distribution nodes
- d** Finds product instance chained to “development” on local and distribution nodes
- f <flavorList>** Described below under “The flavor options”.
- g <chainName>** Finds product instance chained to **<chainName>** on local and distribution nodes; can be a list
- h <host>** Specifies product distribution host to which you are adding the product; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:
 - a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
 - a host and Webserver port number (e.g., *fred.sub.domain:8080*)
 - a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)
- H <flavorList>** Described below under “The flavor options”.
- i** Ignores errors; if command is to be repeated for a list of flavors and errors are generated for one or if download of a component fails, operation continues to the next.
- I** When updating a `ups` directory, unwinds the directory tar file on the local node after transmission (default is to unwind during transmission).
- j** Ignores dependencies, operates just on top level product.
- n** Finds product instance chained to “new” on local and distribution nodes.
- o** Finds product instance chained to “old” on local and distribution nodes.
- O "<flags>"** Sets the value of `$UPS_OPTIONS` to **<flags>**.
- q <qualifierList>** Finds product instance on distribution node with the specified qualifiers (required and/or optional).
- R** Retrieves/operates on only the required dependencies.
- s** Lists what command would do; but does not execute the command
- t** Finds product instance chained to “test” on local and distribution nodes
- v(vvv)** Prints out extra debugging information.
- z <databaseList>** Specifies the local database(s)

The flavor options

Flavor may be specified using **-f**, using **-H** by itself or in combination with any of **-0**, **-1**, **-2**, **-3**, or just using one of **-0**, **-1**, **-2**, **-3**. These options are not valid with each other (except **-H** with a number option).



- f <flavorList>** Finds product instance of specified flavor on local and distribution nodes. If multiple values specified, **upd update** is run once for each flavor. Avoid using **-f** when dependencies are involved, because it can cause flavor mismatches between main product and dependencies.
- H <flavorList>** Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values can be specified, usually of different flavor families. If multiple values specified, **upd update** is run once for each value.

Can be used alone (without an accompanying number option). In this case, the best match instance of the flavor family is picked.

If used with any of **-0**, **-1**, **-2**, **-3**, finds product instance with specified level of that flavor; e.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**. If multiple values specified, **upd update** is run once for each flavor, according to the accompanying number option.
- 0** Specifies flavor as NULL; equivalent to **-f NULL**
- 1** Specifies flavor as OS value up to the generic OS; e.g., on a SunOS machine it is equivalent to **-f SunOS**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX.
- 2** Specifies flavor for product instance on local and distribution nodes as OS value up to the major release; e.g., equivalent to **-f SunOS+5**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.
- 3** Specifies flavor for product instance on local and distribution nodes as most significant OS specification or the full specification; e.g., equivalent to **-f SunOS+5.6**; if given together with **-H IRIX+6.2**, flavor is then specified as IRIX+6.2.

23.12.4 upd update Examples

Before running **upd update**, compare the MODIFIED dates for the product. To do so, run:

```
% ups list -K MODIFIED[:<other_keywords>] [<options>] <product>
  [<version>]
```

on the local node and run **upd list** with similar options on the distribution node.

Update a table file

```
% upd update table_file xntp v3_4 -H SunOS:IRIX:OSF1:Linux
```

UPD updates the table file (the component **table_file** is specified) for the product **xntp v3_4** (and for all its dependencies) for the best match flavors to those listed. If **-f** were used in place of **-H** here, all the dependencies installed would be of the best match flavor to the local machine, and most or all product instances wouldn't work right. Output is provided indicating success or failure (in this case, success):

```
updcmd::updcmd_update - Updating xntp.
upderr::upderr_syslog - successful transfer
ftp://fnkits.fnal.gov//ftp/upsdb/xntp/v3_4SunOS.table -> /tmp/mwmdb/xntp/v3_4.table
upderr::upderr_syslog - successful ups touch xntp v3_4 -f SunOS -q "" -U ""
```



After performing the **upd update**, rerun the **ups list** command to verify the result.

Note: When updating several instances at a time, you can exclude a particular instance from being updated by running **ups touch** on it. See section 22.16 *ups touch* for more information.

23.13 upd verify

The **upd verify** command performs **ups verify** on a distribution database. **ups verify** checks the integrity of the local database files for the specified product(s), and lists any errors and inconsistencies that it finds. See section 22.19 *ups verify* for more information and examples.

23.13.1 Command Syntax

```
% upd verify -h [host] [<ups_verify_options>] [<product>] \  
  [<version>]
```

23.13.2 Options

The **upd verify** command uses all the same options as **ups verify**, plus **-h** (described below). See section 22.19 *ups verify* for the list of options and their descriptions. The difference in the options' functionality is that they apply to the distribution node rather than to the local node.

-h <host> Specifies product distribution host; the default is *fnkits.fnal.gov*. Usually just the plain host name is required, however the **-h** option can always specify any of the following:

- a plain host name; (e.g., *fred.sub.domain*, or just *fred* if *sub.domain* is *fnal.gov*)
- a host and Webserver port number (e.g., *fred.sub.domain:8080*)
- a full URL to the `ups.cgi` cgi script (e.g., `http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi`)

23.14 upp

UPP is a layer on top of **UPD** that can be used to facilitate the update of products on a local **UPS** node as new versions become available on a product distribution node. It has a single command, **upp**, which is controlled by a subscription file. The functions **upp** can be configured to perform on a local node include:

- notifying the client of new and updated products on a specified distribution node
- performing product installations and updates
- installing/updating product dependencies and resolving chains to maintain integrity of parent product
- deleting old product versions



We include this command here to document its syntax. The subscription file, which determines the functionality of the command, is described in Chapter 32: *The UPP Subscription File*.

23.14.1 Command Syntax

```
% upp [-v] <subscription_file_1> [<subscription_file_2>[, ...]]
```

23.14.2 All Valid Options

-v(vvv) Prints out extra debugging information.

23.14.3 upp Examples

```
% upp /path/to/upp.subscription
```

This command executes the subscription file `upp.launch`. No verbose output is requested (no **-v** option). This example is not terribly helpful for two reasons:

- it doesn't show how the subscription file controls **UPP** (beyond the scope of this section), and
- **UPP** is most often used in an automated way, e.g., using **cron**.

We refer you to Chapter 6: *Installing Products Using UPP* and section 10.5.3 *Using UPP to Remove a Product* for examples, and to Chapter 32: *The UPP Subscription File* for detailed information on subscription files.

A better example is to show how to execute the above command using **cron**. We'll add a **cron** job that runs a script we'll call `upp.launch`. This script first sets up **UPD** then runs the **upp** command as shown above:

```
#!/bin/sh
. /usr/local/etc/setups.sh
setup upd
upp /path/to/upp.subscription
```

A sample crontab entry to run the `upp.launch` script every night at midnight might look like:

```
0 0 * * * /path/to/upp.launch
```


Chapter 24: Generic Command Option

Descriptions

This chapter provides an alphabetical listing of **UPS/UPD** options with generic descriptions. More detailed information on a few selected options can be found at the end of the chapter.

In the command reference chapters, Chapter 22: *UPS Command Reference* and Chapter 23: *UPD/UPP Command Reference*, the options supported by each command are listed with command-specific descriptions.

24.1 Alphabetical Option Listing

-?	Prints command description and option usage information to screen Not valid with other options. Note for C shell users: enclose -? in double quotes (e.g., ups declare "-?"); -? is interpreted by sh .
-a	Operates on all instances that match the other options given on command line
-A <nodeList>	Specifies authorized nodes. When declaring an instance, sets the AUTHORIZED_NODES keyword. By default, products can be used from any node which has access to the database; use this option to restrict usage to a limited set of nodes. Note: can also be used inside -G ""
-b <compileFile>	Specifies name of the output file for the ups compile command (COMPILE_FILE keyword) associated with the COMPILE action (see Chapter 37: <i>Use of Compile Scripts in Table Files</i>); .sh or .csh gets added automatically (the file's path is specified with -u)
-B <depProdName>= "<options>"	Specifies options to prepend to the setupRequired line (in table file) for the dependent product <depProdName>
-c	Specifies "current" chain Note: can also be used inside -G ""

-C	<p>For ups declare when initially declaring a product: Prevents execution of the CONFIGURE action</p> <p>For ups declare when declaring a chain: Prevents execution of the corresponding chain action</p> <p>For ups undeclare when removing a product: Prevents execution of the UNCONFIGURE action</p> <p>For ups undeclare when removing a chain: Prevents execution of the corresponding “unchain” action</p>
-d	<p>Specifies “development” chain</p> <p>Note: can also be used inside -G ""</p>
-D "<origin>"	<p>Specifies the product's master source file (This becomes the value of the keyword ORIGIN after any spaces are removed.)</p> <p>Note: can also be used inside -G ""</p>
-e	<p>Sets \$UPS_EXTENDED (to the value 1). See section 24.2.1 -e for more information.</p>
-f <flavorList>	<p>Specifies flavor (operating system[+release]). Multiple values can usually be specified; several UPD commands operate on each listed flavor, however most UPS commands ignore all except first in list.</p> <p>See also -H and the number options -0, -1, -2, and -3.</p> <p>Notes: can also be used inside -G ""; not valid with options -0, -1, -2, or -3</p>
-F	<p>Used only with ups get; prints to screen a list of files that are associated with the product but which are maintained external to the products area (excluding table file)</p>
-g <chainName>	<p>Specifies chain using the chain name (either a standard or user-defined chain name can be used here). Multiple values can usually be specified.</p> <p>Note: can also be used inside -G ""</p>
-G "<options>"	<p>Used by commands that copy or install a product instance (namely ups copy, upd cloneproduct, and upd install), the -G option provides the means to specify target instance identifiers to pass to the internally-run ups declare command.</p> <p>The elements valid for use with -G include <product>, <version> and the following subset of the ups declare options:</p> <p>-A <nodeList>, -c, -d, -D <origin>, -f <flavor>, -g <chainName>, -n, -o, -O "<flagList>", -p "<description>", -q <qualifierList>, -t, -z <databaseList>, -0, -1, -2, -3</p>

<p>-h <host></p>	<p>Specifies distribution host; the default is <i>fnkits.fnal.gov</i>. Usually just the plain host name is required, however the -h option can always specify any of the following:</p> <ul style="list-style-type: none"> • a plain host name (e.g., <i>fred.sub.domain</i>, or just <i>fred</i> if <i>sub.domain</i> is <i>fnal.gov</i>) • a host and Webserver port number (e.g., <i>fred.sub.domain:8080</i>) • a full URL to the <code>ups.cgi</code> cgi script (e.g., <code>http://fred.sub.domain:8080/cgi-bin/some/dir/ups.cgi</code>)
<p>-H <flavorList></p>	<p>Specifies flavor and builds a flavor list for that family starting at the level specified. Multiple values accepted; several UPD commands operate on each listed flavor, however most UPS commands ignore all except first in list.</p> <p>In many commands -H can be used alone (without an accompanying number option). In this case, UPS/UPD finds the best match instance for the specified flavor family.</p> <p>If used with any of -0, -1, -2, -3, UPS/UPD finds the product instance of specified level of that flavor; e.g., -2H IRIX+6.2 is equivalent to -f IRIX+6.</p> <p>Note: can also be used inside -G ""</p>
<p>-i</p>	<p>Ignores errors; allows operation to continue to the next product instance (e.g., if one product dependency can't be downloaded during upd install, it proceeds to the next one)</p>
<p>-I</p>	<p>Unwinds product tar file on the local node after transmission (default is to unwind during transmission); this option allows ftp to "reget" a tar file if network connection is lost during the transfer</p>
<p>-j</p>	<p>Ignores dependencies, operates just on top level product</p>
<p>-J <fileName></p>	<p>Specifies individual file to fetch (for archive database, can only be <code>INSTALL_NOTE</code>, <code>README</code>, or a version or table file)</p>
<p>-k</p>	<p>Prevents execution of unsetup files prior to (subsequent) setup</p>
<p>-K <keywordList></p>	<p>Returns values of specified keywords only; see section 24.2.3 -K for more information</p>
<p>-l</p>	<p>Produces a long listing</p>
<p>-L</p>	<p>Adds the STATISTICS keyword to the version file, thereby instructing UPS to keep statistics on this product instance. A record of the form:</p> <pre>"tcl" "v7_3q" "IRIX" "" "" "berman" "1998-03-13 17.56.54 GMT" "list"</pre> <p>will get added to the file <code>\$PRODUCTS/.upsfiles/statistics/<product></code> each time a UPS command is run on this instance.</p> <p>See section 27.6.3 <i>STATISTICS</i> for more information.</p>

-m <tableFileName>	Specifies table file name
-M <tableFileDir>	Specifies table file directory
-n	Specifies “new” chain Note: can also be used inside -G ""
-N <fileName>	Specifies file to be checked and edited by ups modify
-o	Specifies “old” chain Note: can also be used inside -G ""
-O "<flagList>"	Sets the value of \$UPS_OPTIONS to <flagList>. This is a means of passing information to actions listed in the table file. Note: can also be used inside -G ""
-p "<description>"	Specifies product description Note: can also be used inside -G ""
-P	Requires UPS to rely only on information supplied on the command line to locate the product instance (prevents UPS from searching in a database)
-q <qualifierList>	Specifies required or optional qualifiers; see section 24.2.4 -q for more information Note: can also be used inside -G ""
-r <prodRootDir>	Specifies the product root directory (when declaring a product, sets the value of the keyword PROD_DIR) If <prodRootDir> specifies a relative path, UPS appends it to PROD_DIR_PREFIX in order to construct the entire path. (PROD_DIR_PREFIX is set in the dbconfig file; see Chapter 30: <i>The UPS Configuration File</i>)
-R	Retrieves/operates on only the required dependencies (those listed in table file using setupRequired); ignores the optional ones.
-s	Lists what command would do; but does not execute the command (creates the temp file, but does not source it)
-t	Specifies “test” chain Note: can also be used inside -G ""
-T	Specifies archive file directory or URL (when declaring a product, sets the value of the keyword ARCHIVE_FILE) Note: can also be used inside -G ""
-u <compiledDir>	Specifies the directory for the output file (which is named via the -b option) for the ups compile command associated with the COMPILE action (see Chapter 37: <i>Use of Compile Scripts in Table Files</i>)

-U <upsDir>	Specifies location of <code>ups</code> directory (when declaring a product, sets the value of the keyword <code>UPS_DIR</code>); default value is <code>ups</code> If <code><upsDir></code> specifies a relative path, UPS appends it to the product root directory in order to construct the entire path. Note: can also be used inside <code>-G ""</code>
-v	Prints out extra debugging information. To get progressively more information, use multiple <code>v</code> 's, e.g., <code>-vv</code> , <code>-vvv</code> (up to four).
-V	Does not delete the temporary script files or partially installed products when command finishes; instead lists them on the screen See section 24.2.5 <code>-V</code> for more information on the temporary scripts. Note: can also be used inside <code>-G ""</code>
-w	For <code>ups start</code> , stops the product first, then restarts it
-W	For <code>ups copy</code> , uses environment variables (e.g., <code>\$SETUP_<PRODUCT></code>) to identify dependent product instances for target product (that is, it uses instances that are already setup in preference to what is listed in table file)
-X	For <code>upd install</code> and <code>ups copy</code> : executes the generated <code>ups declare</code> command(s) instead of just printing to screen
-y	For <code>ups undeclare</code> : deletes product root directory, provides confirmation prompt
-Y	For <code>ups undeclare</code> : deletes product root directory, does not provide confirmation prompt
-z <databaseList>	Specifies the local database(s); see 26.1 <i>Database Selection Algorithm</i> for more information Note: can also be used inside <code>-G ""</code>
-Z	Times the command (does not include time for sourcing of temp file for <code>setup/unsetup</code>)
-0	Specifies NULL flavor string; equivalent to <code>-f NULL</code> Not valid with <code>-f</code> or other number options. Can usually be used with <code>-H</code> to specify a single flavor (in this case, NULL). Note: can also be used inside <code>-G ""</code>

<p>-1</p>	<p>Specifies flavor as OS value up to the generic OS (e.g., SunOS); can be used in place of -f <flavor></p> <p>Examples:</p> <ul style="list-style-type: none"> • Used alone on a SunOS machine, it is equivalent to -f SunOS. • If given together with -H IRIX+6.2 on any machine (e.g., -2H IRIX+6.2, it is equivalent to -f IRIX. <p>Not valid with -f or other number options.</p> <p>Note: can also be used inside -G ""</p>
<p>-2</p>	<p>Specifies flavor as OS value up to the major release (e.g., SunOS+5); can be used in place of -f <flavor></p> <p>Examples:</p> <ul style="list-style-type: none"> • Used alone on a SunOS+5 or SunOS+5.6 machine, it is equivalent to -f SunOS+5. • If given together with -H IRIX+6.2 on any machine (e.g., -2H IRIX+6.2, it is equivalent to -f IRIX+6. <p>Not valid with -f or other number options.</p> <p>Note: can also be used inside -G ""</p>
<p>-3</p>	<p>Specifies flavor as most significant OS specification or the full specification (e.g., SunOS+5.6); can be used in place of -f <flavor></p> <p>Examples:</p> <ul style="list-style-type: none"> • Used alone on a SunOS+5.6 machine, it is equivalent to -f SunOS+5.6. • Used alone on a SunOS+5 machine, it is equivalent to -f SunOS+5 (-2 and -3 are equivalent in this case). <p>If given together with -H IRIX+6.2 on any machine (e.g., -3H IRIX+6.2, it is equivalent to -f IRIX+6.2.</p> <p>Not valid with -f or other number options.</p> <p>Note: can also be used inside -G ""</p>

24.2 More Information on Selected Options

24.2.1 -e

UPS_EXTENDED is an on/off type variable. Scripts or the product's table file may contain the UPS_EXTENDED variable to execute "extended" functionality. In order to enable the extended functionality, first **-e** must be set at the time that the product is initially declared, and again in the **setup** command.

For dependencies, first **-e** must be set at the time that the parent product is initially declared, and again in the **setup** command for the parent product.

24.2.2 -H

In order to describe the **-H** option, we first need to define *flavor table* and *flavor levels*:

- a *flavor table* is a list including every level of specificity for a particular OS that you could use to find or declare a product instance. For example, on a SunOS machine of release 5.6, the flavor table generated from **-H SunOS+5.6** is:

```
SunOS+5.6
SunOS+5
SunOS
NULL
ANY
```

- a *flavor level* is a level of specificity for a flavor, e.g., any of the items in the above flavor table.

The **-H** option can be used to run a command "as if" the local machine were of the specified flavor. It can be used alone, in which case **UPS/UPD** uses a best match algorithm to find a matching instance. In contrast, when flavor is specified by **-f**, the command fails if no exact match is found.

The **-H** option can also be used with a number option to specify a particular level of the given flavor family. When used like this, it is equivalent to specifying a flavor via **-f**. E.g., **-2H IRIX+6.2** is equivalent to **-f IRIX+6**.

24.2.3 -K

Two output styles are provided for the commands that print information to the screen: a formatted one that is easy for users to read, and a condensed one for parsing by a subsequent command or a script. Use the **-K** option to request output in the condensed format. The **-K** option requires an argument list specifying which fields to include in the output, for example:

```
% ups list -K product:version:flavor xemacs
```

```
"xemacs" "v19_14" "SunOS+5"
```

The plus sign (+) argument, e.g., **-K+**, is a shorthand for requesting the default fields **product:version:flavor:qualifiers:chain**, for example:

```
% ups list -K+ xemacs
```

```
"xemacs" "v19_14" "SunOS+5" "" "current"
```

Some common keyword arguments used with the **-K** option are:

PRODUCT	product name
FLAVOR	product instance flavor
VERSION	product version
QUALIFIERS	additional instance specification information often used to indicate compilation options used by developer
CHAIN	product instance chain
+	all of the above
DATABASE (or DB)	the UPS database path; useful if more than one on system
DECLARER	logon id of person who declared the instance
DECLARED	date/time that product instance was declared
MODIFIER	logon id of person who modified/updated the instance
MODIFIED	date/time that product instance was modified/updated

The full list of keywords with definitions can be found in section 27.4 *List of Supported Keywords* and in section 22.11 *ups list*.

The condensed format is useful for parsing the output in scripts. A couple of examples are given in section 22.11 *ups list*. It is also convenient for piping **ups list** output to the **grep** command, e.g.,:

```
% ups list -aK+ xemacs | grep OSF1
"xemacs" "v19_14" "OSF1+V3" " " " "
"xemacs" "v19_14" "OSF1+V4" " " " "
"xemacs" "v20_4" "OSF1+V3" " " "current"
"xemacs" "v20_4" "OSF1+V4" " " "current"
```

24.2.4 -q

Products can be declared to the database with qualifiers (see section 1.3.3 *Qualifiers*) using the **-q** option with **ups declare**. To then match any product instance that has been declared with one or more qualifiers (see section 26.2 *Instance Matching within Selected Database*), you must specify the exact set of qualifiers on the command line, again using the **-q** option.



Qualifiers are case-sensitive, and they must be specified on the command line exactly as they appear in the product declaration. Use the **ups list** command to determine how a qualifier was declared.

Required Qualifiers

For instance matching, qualifiers can be entered as *required*, where “required” means that to match an instance, the qualifier *must* appear in its declaration. Required qualifiers are entered in the format **-q <qualifierList>**, for example:

```
% setup -q build gtools v2_1
```

For multiple required qualifiers, the format is `-q <qual_1:qual_2:...>`, for example:

```
% setup -q build:debug gtools v2_1
```

Optional Qualifiers

Alternatively, qualifiers can be requested as *optional*; i.e., they can but don't have to be in the declaration in order to match. If you request a qualifier as optional (e.g., `-q "?qual"`), **UPS** selects an instance declared with that qualifier in preference to an otherwise matching instance without it. Optional qualifier specifications are preceded by a question mark (?) and enclosed in quotes, in the format `-q "<qualifier>"`, for example:

```
% setup -q "?build" gtools v2_1
```

If quotes are not used when ? is included in the string, the shell may return the error "No match".

Mixing Required and Optional

You can list some qualifiers as required and others as optional, where ? precedes only the optional ones, for example:

```
% setup -q "build?debug" gtools v2_1
```

Notice that the colon is not needed between qualifiers if the question mark is used.

24.2.5 -V

A **UPS** command generally executes its internal processes and also looks in the table file for corresponding actions (e.g., `ups declare` looks for ACTION=DECLARE and ACTION=CONFIGURE, by default). If the command finds one or more corresponding actions, the command first translates the functions listed under the actions into shell commands. These commands get written to a temporary script in \$TMPDIR (if \$TMPDIR isn't set, the default is /tmp), and the script is then invoked to execute the shell commands. By default, this script gets deleted. The `-V` option prevents **UPS** from deleting it. This is very useful for debugging table files. Example:

```
% ups declare -V ... myprod v1_0
```

```
INFORMATIONAL: Name of created temp file is /tmp/aaaaakADa
```


Chapter 25: UPS/UPD Command Usage

This chapter describes the syntax for **UPS** and **UPD** commands.

25.1 Syntax

Most **UPS** and **UPD** commands are of the form **ups <command>** or **upd <command>** (the exceptions are **setup** and **unsetup**), and take a variety of command line options and arguments. The standard syntax is:

```
% <command_name> [<options>] [<product> [<version>]] \  
  [, [<options>] [<product> [<version>]]]
```

For example:

```
% ups list -f IRIX+6 xemacs v20_4
```

The allowed arrangements of command line elements are described below.

25.1.1 Order of Command Line Elements

Most commands take two optional positional parameters, the product name and product version, plus any options. The first occurring unflagged element on the line is interpreted as the product name, and the next (if any) is interpreted as the version. With that limitation, the name, version and options can occur anywhere on the command line.

25.1.2 Specifying Version/Chain

When the product version is not specified, and no chain flag is listed, the version associated with the current chain is used.

You cannot specify a chain name (e.g., **new**) in place of the product version or in place of the chain flag (e.g., **-n**) on the command line (this was permissible in the old **UPS**). For example, you can enter:

```
% setup -n foo
```

but you cannot enter:

```
% setup foo new
```



25.1.3 Grouping Option Flags

Option flags can be grouped, for example `-ct` is equivalent to `-c -t`. Only the last option in a grouped list can have an argument, and that argument must follow directly, for example:

```
% ups list -ctf <flavor> <product>
```

25.1.4 Specifying Arguments to Options

Generally, a space is optional between an option flag and its argument. For example, the following two specifications are correct:

```
-f IRIX
```

```
-fIRIX
```

An exception to this rule is the specification of a path starting with tilde (~). There must be a space before the tilde due to `cs` behavior, for example:

```
-r ~user/a/b
```

When an option has multiple arguments, the arguments must be separated by a colon (:)¹, e.g.,

```
% ups list -f IRIX:SunOS:OSF1+V3 proda
```

lists instances of product **proda** for flavors IRIX, SunOS and OSF1+V3. Arguments to options may be enclosed in double quotes, and there may be spaces between them in addition to the required colons. If there are spaces, quotes must be used. A multiple-argument specification can therefore look like any of the following examples:

```
-f IRIX:SunOS
```

```
-fIRIX:SunOS
```

```
-f"IRIX:SunOS"
```

```
-f "IRIX:SunOS"
```

```
-f "IRIX: SunOS "
```

This permits strange looking option lists, for example:

```
-ctfIRIX:SunOS
```

25.1.5 Embedded Spaces in Option Arguments

If an argument has any spaces embedded in it, you must put quotes around it, e.g.,

```
% ups list -q "my compile option" proda
```

Most options you'll find won't have spaces in them. During processing, any arguments that have embedded spaces get packed, with the exception of arguments to the options `-p` (product description) and `-O` (set \$UPS_OPTIONS).

1. A special note for specifying multiple databases: For systems with multiple UPS databases, the databases in \$PRODUCTS may be specified using either colons or spaces as separators (e.g., as `db1:db2:db3` or `db1 db2 db3`) for compatibility with older versions of UPS. But when the `-z` option is used on the command line, only colons are valid as separators between multiple databases (as between multiple arguments for any option); you cannot use spaces.

25.1.6 Invalid Option Arguments

Any argument that can be interpreted as an additional option will be deemed invalid. (This restriction doesn't apply to the `-O` (uppercase `-o`) option.) As an example, the `-1` in the following commands will be interpreted as an invalid flavor:

```
% ups list -f -1 proda
% ups list -f "-1" proda
```

25.1.7 Specifying Multiple Products in a Single Command

A single **UPS** command can be issued for multiple product instances, and in fact for multiple product names, in many cases. The command line elements identifying each product must be separated with a comma (`,`). For example, the following set of three commands:

```
% setup proda
% setup prodb -o
% setup -f IRIX+6.5 prodc
```

can be condensed into:

```
% setup proda, prodb -o, -f IRIX+6.5 prodc
```

The simplest example of this is a command of the form:

```
% ups list proda, prodb, prodc
```

where the comma (`,`) is used simply for separating multiple product names, and the spaces between product specifications are optional. Options are reset when a comma is reached, so even if all the product names in a combined command share the same set of options and arguments, the common elements still need to be listed separately for each product, e.g.,

```
% ups list -f SunOS+5 -z $MY_DB proda, -f SunOS+5 -z $MY_DB
  prodb
```

25.1.8 Multiple Occurrences of Same Option Flag

There can be multiple occurrences of the same flag on the command line and where appropriate, the order is significant and preserved. This is rarely used. As an example:

```
% ups list <product> -c -o -c -g "mychain"
```

lists four chains in the order: current, old, current, and mychain. Here is a more complicated example illustrating option flag grouping as well as repeated options:

```
% ups declare -cf SunOS+5 -Zvvz $MY_DB foo -tnd v2_0
```

The `-v` option requests verbose output; use of multiple `-v`'s increases the level of verbosity. This awkward command declares the product `foo v2_0` as flavor `SunOS+5`, with four chains (current, test, new and development) to the database `$MY_DB`. It also times the command (`-Z`) and requests verbose output (`-vv`).

25.1.9 Use of Wildcards

No wildcard functionality is supported.

25.2 Options

When using the **UPS** and/or **UPD** commands, you must be aware that the options you'll need to specify depend upon your environment. In general, these rules apply:

- **UPS** takes the default flavor as that of the machine on which the command is issued (to the highest specification level possible), or if modifying product instance information, it takes the flavor in the product instance's declaration. This is important to know especially if your database is on a multi-flavor cluster (i.e., includes different operating systems or two or more nodes of the same operating system with different releases). If you wish to specify the flavor differently, use the **-f** or the number options (**-0** through **-3**, described in Chapter 24: *Generic Command Option Descriptions*).
- If your \$PRODUCTS environment variable is set to multiple databases, you may sometimes need to specify the database to use (see section 26.1 *Database Selection Algorithm*) via the **-z** option.
- If there is more than one instance of a particular product installed on your machine, or in the case of a cluster, more than one instance for a single flavor, you can specify the desired instance uniquely by including its version or chain and possibly qualifiers.
- No option flag is associated with product version. The version, if given, must appear after the product name on the command line, although not necessarily immediately after it.
- The number options (**-0**, **-1**, **-2**, and **-3**) cannot be used in combination with each other, or in combination with **-f**. They may be used in combination with **-H** in many commands.

Chapter 26: Product Instance Matching in UPS/UPD Commands

When a **UPS** or **UPD** command is issued, the system must determine which product instance(s) to act upon. This determination is called *instance matching*. This chapter describes the algorithms used for instance matching.

Instance matching for **UPS/UPD** is done within **UPS**. **UPD** commands can find the local **UPD** configuration, which is used to determine the local database for the command; beyond that, it relies on **UPS** to match the instance. A match is determined according to the database, and the product's name, flavor, version and qualifiers. You may specify all these parameters explicitly, in which case no special logic is required; the command will succeed or fail based on the given information. You are not required to specify all the information, however. At a minimum, you must provide the product name explicitly¹. All the remaining information can be defaulted, thus requiring **UPS/UPD** to rely on preset algorithms to match the instance.

26.1 Database Selection Algorithm

26.1.1 UPS

To match a product instance, a **UPS** command must first determine the database in which to search for the product.

Database Not Specified on Command Line

If **-z <databaseList>** is not specified on the command line, **UPS** looks in **\$PRODUCTS**. If **\$PRODUCTS** points to a single database, **UPS** searches only that one. If **\$PRODUCTS** points to multiple databases, then the following applies:

If the command is **ups declare** and it is being used to declare an instance to a database (not to declare a chain to a pre-existing instance) then **UPS**:

- looks at the `updconfig` file for the first database listed in **\$PRODUCTS**,
- finds a matching stanza (the `updconfig` file uses a set of identifiers described in section 31.2 *Product Instance Identification and Matching*),
- and declares the instance to the database specified in the matched `updconfig` file stanza.

If a declaration for the instance already exists in the specified database, the command fails.

1. The commands **ups(d) list**, **ups flavor**, **ups help** and **ups(d) verify** can be used with no product name, version or options.

For all other commands (including **ups declare** when it is being used to declare a chain to a pre-existing instance), **UPS** searches the databases in the order listed in `$PRODUCTS`, and operates on the first matching instance it finds.

Database Specified on Command Line

A database specified on the command line can be any database; it can be `$PRODUCTS` (which is equivalent to leaving off the database specification altogether), it can be one or more databases that are listed in `$PRODUCTS`, it can be one or more databases *not* listed in `$PRODUCTS`, or it can be a combination, e.g., `-z /path/to/somedb:$PRODUCTS`.

In fact, the construction `-z /path/to/somedb:$PRODUCTS` is frequently used (where `/path/to/somedb` may or may not be listed in `$PRODUCTS`). It allows you to give preference to a particular database by placing it first, while still letting **UPS** search the `$PRODUCTS` databases in the usual order for dependencies.

If `-z <databaseList>` is specified on the command line, and:

- `<databaseList>` specifies a single database, **UPS** uses that one.
- `<databaseList>` points to multiple databases, **UPS** searches the specified databases in the order given. It uses the same logic as described in section 26.1 *Database Selection Algorithm*. (Substitute `<databaseList>` for `$PRODUCTS` in the description.)



Note for product dependencies (described in Chapter 16: *Product Dependencies*): If the database is specified on the command line via `-z <databaseList>`, then it gets propagated to all dependencies called using the functions `setupRequired()` or `setupOptional()` (these functions are described in Chapter 34: *Functions used in Actions*). This applies to both top level and lower level dependencies.

26.1.2 UPD

If your **UPS** installation includes only one database, that is the one **UPD** will use (assuming that the **UPD** configuration used by that database points to it, which is generally the case). If there are multiple local databases, the **UPD** command has to determine the database to use. In a nutshell, it:

1) picks a starting database

To pick the starting database, **UPD** first looks to see if the `-z <databaseList>` option is specified on the command line. If so, **UPD** picks the first database listed there. If not, **UPD** picks the first database listed in `$PRODUCTS`.

2) finds the **UPD** configuration file to which the database points¹

3) looks in this **UPD** configuration to see where to install, declare, update, etc., the product (using the `UPS_THIS_DB` location)



If your local **UPS/UPD** installation is particularly complicated, it might be useful to verify that your `$PRODUCTS` variable includes all the databases used by the **UPD** configuration(s), and only those. If not, it is possible to declare products into a database not listed in `$PRODUCTS`, which generally is undesirable.

1. The location of the `updconfig` file is set via the keyword `UPD_USERCODE_DIR` in the database's `dbconfig` file.

26.2 Instance Matching within Selected Database

UPS uses a built-in algorithm based on flavor and qualifier matching in the product's version, chain, and table files to match the instance. Once a match is complete, resulting in a complete product identification, the instance is located and the command executes. The algorithm is described below. **UPD** does not do instance matching, it communicates with **UPS** for this purpose.

26.2.1 Where Does Instance Matching Take Place?

Instance matching takes place in the product's version, chain, and table files. Chain files point to version files which point to table files (this is all described in Chapters Chapter 28: *Version Files*, Chapter 29: *Chain Files* and Chapter 35: *Table Files*), and the instance must be matched at each stage before moving on.

The point at which the process starts depends on what is specified on the command line:

- If a chain is specified on the command line (and no version), then **UPS** first matches the instance in the corresponding chain file. If that chain file doesn't exist, the command fails.
- If no chain or version is specified, then the current chain is assumed by default, and **UPS/UPD** first matches the instance in the `current.chain` file. If this file doesn't exist, the command fails.
- If a version is specified (and no chain), then **UPS/UPD** first matches the instance in the version file. If the specified version file doesn't exist, the command fails.
- Regardless of whether a chain or version was specified, if a table file exists and is specified in the version file, **UPS/UPD** does a final instance matching in the table file. If no match is found, the command fails. If a table file is specified on the command line, **UPS/UPD** bypasses the version and/or chain files, and matching takes place only within the table file.

26.2.2 Flavor Selection

You can specify the flavor on the command line using `-f` or one of `-0`, `-1`, `-2`, `-3`, or `-H` (alone or together with one of `-0`, `-1`, `-2`, `-3`). These options are described in Chapter 24: *Generic Command Option Descriptions*. If you do not specify a flavor, **UPS/UPD** determines the flavor based on the set of rules given in section 26.2.4 *Flavor and Qualifier Matching Algorithm*. In most cases **UPS/UPD** can determine the flavor successfully.

Multiple flavors can be processed for many of the **UPD** commands, and for the **UPS** commands `ups list` and `ups verify`.

26.2.3 Qualifiers: Use in Instance Matching

If the option **-q** is included in the command with qualifiers as arguments, each qualifier can be specified in one of two ways:

- required qualifier (no special characters, e.g., **-q optimize**); command will fail if this qualifier cannot be matched
- optional qualifier (preceded by **?**, e.g., **-q "?debug"**); **UPS/UPD** will try to match this qualifier but command will *not* fail if it cannot be matched

Multiple qualifiers can be specified; mixing required and optional is allowed.

26.2.4 Flavor and Qualifier Matching Algorithm

If a flavor and/or a qualifier list is specified on the command line, **UPS** attempts to match the information exactly. The command fails if an exact match cannot be found (except for optional qualifiers). If no qualifiers are specified, no instance with qualifiers will be selected as a match. If the flavor is not specified or if **-H** is used (alone) to generate a “host” flavor list, **UPS** selects the first matching instance it encounters according to this algorithm:

- 1) That instance with a flavor exactly matching the machine’s or host `<OS_type>+<OS_version>` and matching qualifiers, if any.
- 2) That instance with a flavor exactly matching the machine’s or host `<OS_type>` and the OS version specification which is the longest first substring of `<OS_version>`, and matching qualifiers, if any.
- 3) That instance with a flavor exactly matching the machine’s or host `<OS_type>` and having no release specification, and matching qualifiers, if any.
- 4) That instance with a flavor of `NULL`, and matching qualifiers, if any.

The selection fails if none of the above can be met within the determined database.



Be aware that the flavor `ANY` is allowable in a table file, and as expected, it will serve as a match for any flavor, even if the actual flavor appears later in the file. A flavor list generated by **-H** includes the value `ANY`.

Glossary

This glossary defines terminology as it is used in the context of **UPS** and **UPD** v4.

action

Also called a **UPS** action. Actions are used in table files to group together functions that **UPS** must perform when a particular command is issued. An action consists of an ACTION=VALUE keyword (e.g., ACTION=SETUP) plus any functions listed underneath it.



active product instance

The product instance that is currently setup. The *active instance* may be different than the *current instance*.

archive UPS database

A **UPS** database on a product distribution node in which the **UPS** product instances are stored in archive format (e.g., tar, gzip), available for downloading to a user node. Also called a *distribution database*.

bootstrap

(In this manual, we discuss bootstrapping the **CoreFUE** product, which includes **UPS**, **UPD** and **perl**.) Install **UPS/UPD** on a machine on which no prior versions of these products are installed.

build

The process by which a distributable instance of a software product is constructed. The build procedure results in a unique combination of product name, version, flavor, and qualifiers. The actual process varies by product and by developer. It can simply consist of a set of copy commands, or be as sophisticated as generation of executables from a master source library of the software.

chain

A chain is a **UPS** database entry (in a chain file) that points to a declared product instance, tagging the product instance according to its release status (e.g., current, test). Chains allow users to specify the version of a product according to its status, rather than by its version number. The defined chain names are: *current*, *test*, *development*, *new*, and *old*. Their corresponding options (or flags) used in commands are: **-c**, **-t**, **-d**, **-n**, **-o**. The **-g <chainName>** option allows definition of an arbitrary chain name.

Chains are set by the **ups declare** command; hence the term *declare a product instance as current*.

chain file

Chain files reside in the product-specific directory under the **UPS** database directory, and maintain the chain information. Chain files are named according to the chain name, and end with *.chain*, e.g., *current.chain*. A chain file's contents is simply the list of the product instances (specified via sets of keyword/value pairs) that have been declared with that chain.

cluster

For the purposes of this document, a cluster is set of CPU nodes which share one or more **UPS** databases and product areas. Generally the nodes of a cluster also share (at least) login areas.

configure a product instance

For any product instance that requires configuration, an **ACTION=CONFIGURE** line is provided in its table file, with functions listed beneath it. In **UPS** *configuring a product instance* means executing these functions by issuing the **ups configure** command with appropriate options. This happens by default when a product is declared, otherwise it can be run manually. The functions perform all the configuration needed for the product to run, minus that which requires input from the installer (see *tailor a product instance* and *INSTALL_NOTE* for that portion).

coreFUE

A bundle of **UPS**, **UPD** and **perl**, the core pieces of the Fermi UNIX Environment.

current instance (of a product)

A product instance that is declared as current in the database (i.e., to which the chain “current” points). The current instance of a product is the default for **UPS** and **UPD** commands when no version or chain is specified. For a given product, there may be one current instance each for several flavor/qualifier pairs.

daemon process

A background process that is configured to start up automatically on a system at boot time and to stop at shutdown.

database

See *UPS database*.

database configuration file

The **UPS** database configuration file contains system-specific information that customizes the **UPS** installation on a node or cluster. If it exists, it must reside under the database directory in the file `/path/to/ups_database/.upsfiles/dbconfig`.

declare a product instance to UPS

The **ups declare** command makes a product instance known to the **UPS** database and accessible by **UPS**. Declaration does not by itself make the product instance usable since any product requirements (and often other conditions) must also be satisfied, but declaring the product instance is a prerequisite for use (unless you’re using **UPS** products without a database).

declare a product instance current

Declaring a product instance as “current” essentially tags it as the default instance (when its flavor/qualifiers are matched). The declaration creates a current chain file or chain file entry that points to the version file for the instance. Product instances can also be declared as test, development, new or old, or as a user-defined chain for easy access.

declared product instance

An instance of a product which has been declared to a **UPS** database.

default function

The functions (as listed in section 34.3 *Function Descriptions*) that a **UPS** command completes (in addition to its internal processes) if no corresponding **ACTION=COMMAND** keyword line is found in the matched table file, or if the function **doDefaultFaults([<ACTION>])** is listed under the corresponding **ACTION=COMMAND** keyword line. Only the commands **setup** and **unsetup** actually have default functions.

dependencies

Additional products that must be installed, declared, and setup to ensure the successful operation of a given product or to enable special features within it. When a product instance is setup, its dependencies also get setup by default.

distribution database

A **UPS** database in which **UPS** product instances are available for distribution to user nodes. A distribution database may be in archive or live format. The default distribution database at Fermilab is **KITS** which is maintained by the Computing Division on the node *fnkits.fnal.gov*.

distribution node

This term is used in **UPD** to refer to the node on which **UPS** products are stored and available for distribution to user nodes. A distribution node contains a distribution **UPS** database (can be live or archive) and a distribution products area, and runs **UPS**, **UPD**, a Web server and an **FTP** server (preferably **WU-FTP**). It is sometimes called a *server node*.

It is possible to maintain a distribution database on one machine running **UPS** and **UPD** and a Web server, and maintain the corresponding distribution products area(s) on a different one running an **FTP** server, if the machines share a file system.

end user

Anyone who uses **UPS** products, but does not install, update, maintain, or develop them.

FermiTools

FermiTools are Fermilab-developed software packages that are believed to have general value to other application domains, and thus have been made publicly available in a special subdirectory of **KITS** via anonymous **FTP** and **www**. They do not require **UPS**. Installation and use instructions come with each product.

Fermi UNIX Environment (FUE)

FUE started as a project for providing a cross-department, cross-division structure for the proposal, discussion, design and implementation of all things that affect the user when operating in a UNIX environment at Fermilab. Currently it consists of scripts and programs that form a uniform UNIX environment, standards documents, and the **UPS** suite of tools (see <http://www.fnal.gov/cd/FUE/>).

flavor

To indicate the operating system (OS) dependency of a product instance, we use the term *flavor*. This extra term allows us to differentiate by operating system, and optionally OS version, while maintaining the same product name and version number for separate instances. Some products do not require customizing for the different operating systems (typically those without compiled code), but most do and therefore come in several flavors.

flavor table

A list of a machine's flavor including every level of specificity that you could use to find or declare a product instance. For example, on a SunOS+5.6 machine, the complete flavor table reads:

```
SunOS+5.6
SunOS+5
SunOS
NULL
ANY
```

FTP server node

As regards **UPD**, this node contains **UPS** product instances (and files associated with them) that may be downloaded to a user node, and it runs an **FTP** server. Usually it is the same node as the Web server node, and called simply the *server node* or the *distribution node*.

FUE

See *Fermi UNIX Environment*.

fullFUE

A bundle of **coreFUE** plus the pieces which are strongly recommended for on-site systems: **systools**, **shells** and **futil**.

function

A **UPS**-defined entity used in table files that executes an operation within an action. The supported functions are listed in section 34.3 *Function Descriptions*. One or more functions always follow an ACTION=VALUE keyword line.

A function is specified in a shell-independent manner, but contains enough information to allow it to be transformed into a **sh** or **csh** family command (e.g., **sourceRequired()**, or **execute()**), or to be interpreted directly by **UPS** (e.g., **writeCompileScript()**).

install a product instance

Copy a product instance to a local system from another location (usually from a distribution node) and perform the necessary steps to make it work.

INSTALL_NOTE

A file that describes procedures that the installer must perform manually to complete the installation of a product. This file is provided by the product developer as needed.

instance

See *product instance*.

internal processes (or internals)

The set of processes that a **UPS** command completes, regardless of the contents of the product instance's table file. The internal processes are driven by the command line parameters and options, and relevant environment variables.

keyword

Keywords are used in the **UPS** database files. They are essentially parameters to which values must be assigned. The supported set of keywords listed in section 27.4 *List of Supported Keywords* collectively contains the information **UPS** requires for managing a **UPS** installation and all its **UPS** products. Some of the keywords can be used in all the **UPS** product management file types, others are restricted to certain file types.

keyword value

The value assigned to a keyword in one of the **UPS** database files.

KITS

The name of the **UPS** product distribution database on the central product distribution node at Fermilab, *fnkits.fnal.gov*. The location of the **KITS** database is `/ftp/upsdb`. **UPS** products are stored in the corresponding product area, `/ftp/products` (symlinked to `/ftp/KITS`), as tar files, generally. **UPD** commands access the **KITS** database and products area by default.

live UPS database

A **UPS** database in which the **UPS** product instances are unwound, i.e., not stored in archived format (e.g., tar, gzip).

local UPS database

A live **UPS** database on a local node. For user nodes, a database in which **UPS** product instances are declared and available to be accessed and used.

local user node

See *user node*.

make

The UNIX **make** utility is a tool for organizing and facilitating the update of executables or other files which are built from one or more constituent files. See *UNIX at Fermilab* or a standard UNIX reference text for more information.

Makefile

First, see *make* above. A Makefile is a blueprint that you design and that **make** uses to create or update one or more target files (usually executables) based on the most recent modify dates of the constituent files. See *UNIX at Fermilab* or a standard UNIX reference text for more information.

operating system (OS)

A control program for a computer that allocates computer resources, schedules tasks and provides the user with a way to access the resources. See document *DR0010* in the Computing Division Web pages for the latest information on supported UNIX operating systems at Fermilab.

operating system version (OS version)

Like other software, an operating system gets fixed and enhanced periodically, and is released by the vendor with a new version number (e.g., IRIX 5.1, IRIX 5.2). Sometimes **UPS** products must be changed to continue to work properly under a new operating system version.

operating system type (OS type)

The name of the basic operating system, without release number, as returned by the command **ups flavor -2** (for example IRIX or SunOS).

overlay

An overlaid product gets distributed and maintained in the product root directory of its main product. The set of products overlaid on a main product is collectively referred to as *the overlay*.

parent product

A dependency's *parent product* is that for which it is a dependency. A product may have multiple parent products.

platform

Platform technically refers to the machine type (hardware) of a computer system. However, since until quite recently in the UNIX world there has been a near-perfect correspondence between hardware platform and OS type (e.g., Digital Alphastations run OSF1), sometimes platform is used loosely to refer to the OS type. This correspondence is changing as Linux can be run on PC, Digital, Sun and IBM hardware.

process an action

UPS converts the shell-independent functions listed underneath an ACTION keyword line in a table file into code appropriate to the shell, and writes the output to a temporary file. This is call processing an action.

product

See *UPS product*

product developer

A person who develops and maintains software products, and makes them available for distribution by installing and declaring them to the **KITS** or other distribution database. Sometimes called a product maintainer.

product installer

A person who downloads **UPS** products from a distribution node (through **UPD**, **UPP** or **FTP**), installs them on a local system, and declares them to a local **UPS** database (often the local system administrator acts as the product installer).

product instance

The term *product instance*, or just *instance*, is used to represent a copy of a product, namely a unique combination of product name, version, flavor and qualifiers within a **UPS** database. For a given product, multiple instances may exist in the database to allow users a choice of version and/or flavor/qualifier pair. A product instance may be chained; hence the term “the current instance of a product”.

product name

The name of a **UPS** product as it appears in its **UPS** database files.

product root directory

The directory in which a product instance (i.e. its executables) and (optionally) its associated files reside. The product instance generally has a directory structure of its own, starting at this root directory. Each instance of a product has a separate product root directory.

product user

See *end user*.

product version

The net result of any change to an existing product is that a new *version* of the product is created; it is still the same product, but it will usually run a little differently. The versions of a product are tracked by version numbers, e.g., v1_0, v1_1, etc. **UPS** allows for multiple versions of a given product to be accessible concurrently to end users.

PRODUCTS (or \$PRODUCTS)

The environment variable that points to the **UPS** database(s) on your system. If multiple **UPS** databases exist, \$PRODUCTS can be reset in your login files to a colon-separated list of databases.

<PRODUCT>_DIR (or \$<PRODUCT>_DIR)

PRODUCT here is the name of a product in upper case (e.g., EMACS_DIR). This is the environment variable that points to the product root directory of the active instance of a particular product; it gets set when the **setup** command is run.

qualifier

The product developer may include information about options used at compilation time (e.g., *debug* or *optimized*) or other qualifying information for easy identification of special compilations. This information is declared in the form of *qualifiers*. Qualifiers, when present, are part of the unique instance identification along with product name, version and flavor.

read-only variable

UPS sets several read-only variables that can be used in functions in table files. Many of them correspond to keywords set in the **UPS** configuration file. There is another set of read-only variables available for use in setting location definitions in the **UPD** configuration file.

root directory for product

See *product root directory*.

setup

Each installed, declared **UPS** product instance requires that the **setup** command be issued prior to use (unless it is a dependency of one that is already setup). **setup** performs the necessary operations in your login environment to make an installed, declared product accessible to you. Typically, the operations include modifying environment variables or adding to your \$PATH. Any dependencies defined for the product get setup by default at the same time.

table file

Table files contain non-system-specific and non-shell-specific information that **UPS** uses for installing, initializing, and otherwise operating on product instances. That is, information pertinent to one or more product instances, independent of the installation machine. Table files are provided by the product developer as needed.

tailor a product instance

Tailoring is the aspect of the product implementation that requires input from the product installer (e.g., specifying the location of hardware devices for a software driver package). If the product requires tailoring, a file is usually supplied in the format of an interactive executable (script or compiled binary), and it is run by issuing the **ups tailor** command with appropriate options. To *tailor a product instance* means to run this action, and hence, run the file.

tar

The **tar** (tape archive) utility can create, add to, list, and retrieve files from an archive file.

tar file

A tar file is in archived format, and must be unwound for use. **UPS** products are generally stored in `KITS` as tar files.

unknown command handler

A **UPS** feature that allows user-defined actions (e.g., `ACTION=XYZ` followed by **UPS**-supported functions) in table files that can be run via a corresponding **UPS**-style command (e.g., **ups xyz [<options>] <product> [<version>]**)

unsetup

unsetup generally undoes the changes to the user's software environment made by **setup** in order to make the product no longer available for use. Any dependencies get deactivated automatically at the same time by default.

UPD - Unix Product Distribution

A companion product to **UPS** which provides the functionality for uploading/downloading products between local systems and product distribution servers.

UPD commands

Any of the commands supported by **UPD**. They are listed and described in Chapter 23: *UPD/UPP Command Reference*. These include commands to retrieve **UPS** products or certain individual files or directories from a distribution database, and commands to manage products within a distribution database.

UPP - Unix Product Poll

A layer on top of **UPD** that allows a client to request notification of changes in a distribution node database and to download pre-specified products. **UPP** can be automated. This is a useful tool for keeping abreast of changes/enhancements to your favorite products.

UPS - Unix Product Support

UNIX Product Support (**UPS**) is a software support toolkit which provides a methodology for creating/managing all the UNIX products provided and/or supported by the Computing Division, and a uniform interface for accessing these products. **UPS** is itself a product that must be installed on any machine that will be used to run other **UPS** products.

UPS has two parts: one or more databases which function as a central repository of information about the products, and a set of procedures/programs to manipulate the database(s).

UPS action

See action.

UPS commands

Any of the commands supported by **UPS** to manage products in a **UPS** environment. They are listed and described in section Chapter 22: *UPS Command Reference*.

UPS database

A directory that functions as a repository of information about all the installed, accessible **UPS** product instances on a system. **UPS** allows multiple installed and declared instances of each product. The database contains files for each product which store pointers to and information about the declared instances of the product.

ups directory (or ups subdirectory)

A directory that may contain miscellaneous important files for a product instance; e.g., its table file, scripts that the table file needs to execute, and so on. This directory may reside anywhere; it often resides directly under the product instance's root directory. Not all products have `ups` directories.

UPS product

Software products distributed and managed by the **UPS** system are called **UPS** products. **UPS** products include Fermilab-written programs, a wide range of public domain software, and a host of third party licensed (proprietary) products. **UPS** products are available for distribution in the `KITS` database on *fnkits.fnal.gov*.

user node

A node from which users can run **UPS** products; usually contains a live local **UPS** database and locally-installed products.

version

For a product see *product version*; for an operating system see *operating system version*.

version file

A version file contains system-specific information for each instance of a **UPS** product. One *version file* must exist in the product-specific directory under the **UPS** database directory for each version of a product that is declared to the **UPS** database. The name of the version file is the version number followed by `.version`, e.g., `v2_2.version`.

Web server node

As regards **UPD**, this node contains one or more distribution databases and runs a Web server, and **coreFUE**. Usually it is the same node as the **FTP** server node, and called simply the *server node* or the *distribution node*.

Index

Symbols

"-?" option 2-1, 10-1
+ argument for -K option 2-3
.upfiles directory 1-6
.upsfiles directory 1-6
/etc/init.d directory 14-5
/etc/rc*.d directories 14-5
/usr/local/ area
 Fermilab policy regarding use of 15-2, 15-3
@ symbol 27-8
 use with keywords 22-46
_UPD_OVERLAY keyword 16-7, 27-11
 description 27-8

Variables

\$_<PRODUCT>_DIR variable 34-18
 as set during setup 22-5
 description 22-5
\${DASH_PROD_FLAVOR} read-only variable 31-4
\${DASH_PROD_QUALIFIERS} read-only variable 31-5
\${PROD_DIR_PREFIX} read-only variable 31-5
\${PRODUCTS} read-only variable
 comparison to PRODUCTS env variable 34-19
 description 34-19
\${SUFFIX} read-only variable 31-5
\${UPD_USERCODE_DB} read-only variable 3-4
\${UPD_USERCODE_DIR} read-only variable 3-4
\${UPS_BASE_FLAVOR} read-only variable 31-4
\${UPS_COMPILE} read-only variable
 description 34-19
\${UPS_EXTENDED} read-only variable
 description 34-19
\${UPS_OPTIONS} read-only variable
 description 34-19
\${UPS_ORIGIN} read-only variable
 description 34-19
\${UPS_OS_FLAVOR} read-only variable
 description 34-19
\${UPS_PROD_DIR} read-only variable
 description 34-19
\${UPS_PROD_FLAVOR} read-only variable 31-4
 description 34-19
\${UPS_PROD_NAME} read-only variable 31-4
 description 34-19
\${UPS_PROD_QUALIFIERS} read-only variable 31-4
 description 34-19

\${UPS_PROD_VERSION} read-only variable
 description 34-19
\${UPS_THIS_DB} read-only variable
 description 34-19
\${UPS_UPS_DIR} read-only variable
 description 34-20
\${UPS_USERCODE_DB} read-only variable 31-4
\${UPS_USERCODE_DIR} read-only variable 31-4
\${UPS_VERBOSE} read-only variable
 description 34-20
\$PATH variable 1-10, 2-10, 22-11
\$PRODUCTS variable 1-6, 1-10, 25-4
 as used in UPD commands 26-1
 as used in upd install 5-2
 comparison to read-only \${PRODUCTS} 34-19
 for multiple databases 25-2
 use in database selection 26-1
 use with private database 11-9
 with AFS database 12-4
\$SETUP_<DIR> variable 34-18
 as set during setup 22-5
\$SETUP_<PRODUCT> variable
 description 22-5
 use with unsetup command 22-6, 22-11
\$SETUP_UPS variable 1-10
\$TEMPDIR variable
 use with upd addproduct 17-1, 23-7
\$UPS_DIR variable 1-10
\$UPS_EXTENDED variable
 as set by -e option 24-2
\$UPS_EXTRA_DIR variable 12-5
\$UPS_OPTIONS variable
 as set by -O option 24-4
\$UPS_SHELL variable 1-10

"@" Keywords

@COMPILE_FILE keyword 22-47, 27-9
@PROD_DIR keyword 22-48, 27-9
@TABLE_FILE keyword 22-48, 27-10
@UPS_DIR keyword 22-48, 27-11

A

access.conf file 20-11
accessing a UPS product 2-8, 22-5

- accounts
 - for managing distrib node 20-3
 - for product installation 11-1, 11-2
 - ftp 20-3, 20-8
 - separate by product category 11-2
 - the products account 11-1
 - upadmin 20-3, 20-5, 20-12, 21-6
 - wwwadm 20-3, 20-4, 20-7, 20-8
- ACTION keyword
 - "unchain" names as values 33-3
 - chain names as values 33-3
 - description 27-4
 - detailed 31-5, 33-1
 - UPS command as keyword value 33-1
 - use in table files 34-1
 - user-defined values 33-3
- actions
 - "unchain" name as keyword value 33-3
 - and "unactions" 33-2
 - called by other actions 33-4
 - chain name as keyword value 33-3
 - examples 34-18
 - functions used in 34-1
 - overview 31-5, 33-1
 - processing of 24-9
 - reference 33-1
 - undoing chains in table files 33-3
 - undoing reversible functions 33-2
 - UPS commands used as 33-1
 - use in table files 33-1
 - use in updconfig 31-5
 - use with "unknown" commands 33-3
- add chain to product on distrib node 17-7, 23-33
- add product to distrib node 17-3, 23-3, 23-8
 - using template_product 18-6
- add product to KITS 17-3, 23-3, 23-8
 - special product registration 17-3
- add table file to distrib node 17-5
 - update for existing product 17-6
- add ups directory to distrib node
 - update to existing product 17-6
- addAlias function
 - description 34-2
- AFS
 - \$PRODUCTS variable 12-4
 - \$UPS_EXTRA_DIR variable 12-5
 - configuring local database 12-2
 - installing into local database 12-5
 - installing into local products area 12-4
 - installing product into AFS product area 8-3
 - local configuration options 12-1
 - local FUE initialization files 12-3
 - products requiring special privileges 12-6
 - providing access to AFS products 12-1
 - updating /usr/local/bin 12-6
 - upsdb_list file 12-2
 - using AFS UPD and installing locally 8-2
 - using local database with 12-1, 12-2
- AFS database
 - use with local database 5-3
- aliases defined by UPS 1-10
- announcement of new/updated product 17-10
- anonymous FTP 7-5
 - download files from fnkits 7-2

- apache product
 - for distrib node web server 20-5, 20-10
- apropos command 38-3
- ARCHIVE_FILE keyword 22-47
 - as set by -T option 24-4
 - description 27-4, 28-2
- AUTHORIZED_NODES keyword 22-47, 30-1
 - as set by -A option 24-1
 - description 27-4, 28-2
- autostart
 - configuring UPS to allow 14-1
 - control files 14-3
 - permissions 14-4
 - disabling 14-5
 - installing product for 14-2
 - START action 14-3
 - start script example 36-4
 - STOP action 14-3
 - stop script example 36-5
 - TAILOR action 14-3
 - ups script 14-1
 - ups_shutdown script 14-1, 14-2
 - ups_startup script 14-1, 14-2

B

- bin directory of product 16-1, 16-3, 16-5, 18-4, 19-1
 - description 15-6
- bootstrapping CoreFUE
 - bootstrap script 13-1, 13-5
 - config.custom file 13-2
 - configurator script 13-2
 - customizing configuration 13-3
 - log file 13-5
 - predefined configurations
 - for NT 13-2
 - for UNIX 13-1
 - running the procedure 13-5
 - sample customization 13-4
 - space requirements 13-1
 - stage1.sh file 13-1, 13-5
 - stage2.sh file 13-5
 - user defined configurations 13-2
 - user-customized configuration 13-2

C

- catman directory 15-7
- CATMAN_SOURCE_DIR keyword 22-47
 - description 27-4
- CATMAN_TARGET_DIR keyword 22-47, 30-1
 - description 27-4
- CD-ROM
 - product distribution 20-14
 - setup product directly from 22-7
- chain
 - adding product to distrib node 17-3, 23-7
 - as action in table files 33-3
 - change (on declared instance) 10-7
 - current 1-4
 - declare at product declaration 3-6, 10-2
 - declare to installed instance 10-4

- definition 1-4
- development 1-4
 - new 1-4
 - old 1-4
 - remove and add new 10-7
 - remove from instance 10-6
 - specification in command 25-1
 - test 1-4
 - usage 1-5
 - use in instance matching 26-3
 - user-defined 1-4
- chain files 1-6, 22-79, 29-1
 - and product removal 10-7
 - creating 29-1
 - description 29-1
 - examples 29-3
 - information storage format 29-1
 - instance matching within 26-3
 - keywords 29-1
 - overview 27-1
- CHAIN keyword 22-47
 - description 27-4, 29-2
- chain names 1-5
- chain options 1-5
- change a chain 10-7
- change product chain on distrib node 17-7
- command defaults 1-8
- command output formats for ups list 24-7
- command syntax 1-8
 - description 25-1
- comment solicitation INT-5
- COMMON: keyword 35-3
 - description 27-4
 - use in table files 35-3
 - use in updconfig file 31-2
- COMPILE action 37-1
- compile script 37-1
- COMPILE_DIR keyword 22-47, 27-9
 - description 27-4, 28-2
- COMPILE_FILE keyword 22-47, 27-9
 - as set by -b option 24-1
 - description 27-4, 28-2
- config.custom file 13-2
- configurator script 13-2
- configure a product instance 3-9, 22-13
 - in AFS space 8-5
- CONFIGURE action 10-8, 22-80, 36-1
- configure script 36-1
 - for prebuilt binaries 16-5
- configuring distribution node 20-1
- conventions, notational INT-3
- copy a product declaration 22-19
- CoreFUE
 - and AFS 12-1
 - bootstrapping 13-1
 - components 12-4, 12-5, 13-1
 - customizing configuration 13-3
 - local installation on AFS machine 12-4
 - predefined configurations
 - for NT 13-2
 - for UNIX 13-1
 - running the bootstrap procedure 13-5
 - sample bootstrap customization 13-4
 - space requirements 13-1
 - user defined configurations 13-2

- courtesy links to initialization files 1-9
- create a database
 - checklist for preparation 11-9
 - on machine running AFS 12-2
- cron
 - use to automate UPP 4-3, 6-4
- CURRENT action 36-3
- current chain 1-4
 - as default 1-8
- current script 36-3
- CVS 17-9
 - use with template_product 18-8
- CYGWIN
 - bin directory 11-8
 - perl version 11-7
 - UPS/UPD installation issues 11-7

D

- database (See UPS database)
- database configuration file (See UPS configuration file)
- database files
 - chain files 29-1
 - included comments 27-3
 - keywords 27-1
 - location 11-6
 - ownership 11-3
 - permissions 11-3
 - pointers to directories 11-6
 - syntax 27-3
 - UPD configuration file 31-1
 - UPP subscription file 32-1
 - UPS configuration file 30-1
 - version files 28-1
- database on distrib node
 - file permissions 20-7
 - host-based access restriction 20-6
 - user-based access restriction 20-6
- database selection algorithm 5-2, 26-1
- database specification in commands 25-4
- dbconfig file (See UPS configuration file)
- dbconfig.template file 30-1
 - listing 30-2
- declare a chain to an instance 3-6, 10-2, 22-21
- declare a product 3-5, 10-1, 22-21
 - after download via FTP 3-5, 10-1
 - as part of installation 5-1
 - declare chain at same time 3-6, 10-2
 - node/flavor-specific functions present 10-4
 - specifying ups dir and table dir 3-5, 10-2
 - to local database 7-4
- DECLARED keyword 10-6, 22-47
 - description 27-4, 28-2, 29-2
- DECLARER keyword 10-6, 22-47
 - description 27-4, 28-2, 29-2
- defaults for UPS/UPD commands 1-8
 - Also see command reference chapters
- delete product component from distrib node 17-8
- delete product from distrib node 17-8
 - using template_product 18-8
- dependencies
 - and unsetup command 22-11
 - conflict resolution 35-4

- cross-database support for 1-5
- database selection for install 5-3
- definition 1-5
- finding them for a product 2-7
- list using ups depend 2-7
- multiple levels of 1-5
- non-UPS products 35-4
- on distribution node, list using upd depend 4-5
- order of product setups 35-5
- setupOptional function in table file 35-4
- setupRequired function in table file 35-4
- dependency matching 26-2
- DESCRIPTION keyword 22-47
 - description 27-4, 28-2, 29-2
- determine if product update needed
 - using upd install -s 10-13
 - using upd update -s 10-13
 - using upp 10-13
- development chain 1-4
 - use during product development 16-2
- distributing UPS products
 - announcement policies for new products 17-10
 - overview 17-1
 - to KITS (checklist) 19-3
 - to KITS (using template_product) 19-3
- distribution node
 - ~ftp area 20-4
 - access restrictions on database
 - host-based 20-6
 - user-based 20-6
 - configuration and management 20-1
 - configure and manage 20-1
 - fnkits.fnal.gov 3-2, 7-2
 - FTP server 20-1
 - configuration 20-7
 - KITS database (on fnkits.fnal.gov) 3-2
 - limiting product distribution 20-11
 - nodes other than fnkits 7-4
 - option_list product description 20-12
 - reporting on FTP and Web accesses 20-10
 - response to upd addproduct command 20-2
 - response to UPD commands 20-1
 - response to upd install command 20-2
 - response to upd modproduct command 20-2
 - restrict downloads from database 20-11
 - restrict uploads to database 20-11
 - updconfig pre and postdeclare actions 20-10
 - user accounts 20-3
 - web server 20-1
 - configuration 20-5
- doc directory 15-7
- documentation for products 15-7
- doDefaults function
 - description 34-3

E

- editing database files 10-11
- END: keyword 35-3
 - description 27-4
 - use in table files 35-3
 - use in updconfig file 31-2

- envAppend function
 - description 34-3
- environment
 - and usage of command options 25-4
 - changes made by UPS 1-10
 - initializing for UPS 1-9
- envPrepend function
 - description 34-4
- envRemove function
 - description 34-4
- envSet function
 - description 34-5
- envSetIfNotSet function
 - description 34-5
- envUnset function
 - description 34-5
- examples directory 15-7
- exeAccess function
 - description 34-6
- exeActionOptional function
 - description 34-6
 - use to call another action 33-4
- exeActionRequired function
 - description 34-6
 - use to call another action 33-4
- execute function
 - description 31-6, 34-7
 - use in dbconfig 31-6

F

- Fermi UNIX Environment
 - initializing 1-9
- FermiTools INT-2, 4-6, 7-1, 7-2, 21-3
- FILE keyword 30-1
 - description 27-5, 28-2, 29-2
- file ownership
 - considerations 11-3
 - database files 11-3
 - product files 11-3
- file permissions
 - configuring UPD to set (product files) 11-2
 - database files 11-3
 - extra security 11-3
 - unwound tar files 11-2
- file system semantics
 - and group ids 11-2
 - Berkeley 11-2
 - setting 11-2
 - System V 11-2
- fileTest function
 - description 34-7
- flavor
 - ANY, as used in flavor matching 26-4
 - definition 1-3
 - NULL 1-3
 - specification in KITS 1-3
- FLAVOR keyword 22-47
 - description 27-5, 28-2, 29-2
 - value ANY 35-3
- flavor levels 2-2, 24-7
- flavor of machine, determining 2-1, 22-35

- flavor specification
 - (-f, -H and number options) 1-3
 - use in instance matching 26-3
- flavor table 24-7
 - definition 2-2, 22-36
- flavor.products file 14-3, 14-5
 - permissions 14-4
- fnalonly products 21-3
- fnkits.fnal.gov distribution node 4-1
 - adding products to 23-8
 - anonymous FTP for downloading products 7-1, 7-2
 - config file locations 21-6
 - database location 21-6
 - directory hierarchy 4-6
 - FermiTools 4-6, 7-1, 7-2
 - FTP server log file 21-7
 - ftpgroups file 21-6
 - KITS database 3-2
 - KITS product categories 21-3
 - product pathnames for FTP access 4-7, 4-8
 - product permissions 4-6
 - proprietary products 4-8
 - registration for downloading products 3-2, 7-2
 - server maintenance 21-6
 - using FTP to download products 7-1
 - web server log file 21-7
- formatted ups list output 22-45
- FTP
 - declare product after download 3-5, 10-1
 - downloading product components 7-1
 - product installation 7-1, 7-2, 7-5
- FTP server
 - access file 20-11
 - log file on fnkits 21-7
 - log searching 20-13
 - on distrib node 20-1
- ftpaccess file 20-7, 20-11
- ftpgroups file 21-6
- ftpweblog product 20-10
- FUE initialization files
 - courtesy links to 12-3, 12-5, 12-6
 - for use with AFS 12-3
- functions
 - addAlias 34-2
 - case (in)sensitivity of 34-1
 - doDefaults 34-3
 - envAppend 34-3
 - envPrepend 34-4
 - envRemove 34-4
 - envSet 34-5
 - envSetIfNotSet 34-5
 - envUnset 34-5
 - examples 34-18
 - exeAccess 34-6
 - exeActionOptional 34-6
 - exeActionRequired 34-6
 - execute 31-6, 34-7
 - fileTest 34-7
 - overview 34-1
 - pathAppend 34-8
 - pathPrepend 34-8
 - pathRemove 34-9
 - pathSet 34-9
 - preprocessing via compile script 37-1
 - prodDir 34-9
 - reference 34-1
 - reversible 33-2, 34-1
 - setupEnv 34-10
 - setupOptional 34-10
 - setupRequired 34-10
 - sourceCompileOpt 34-11
 - sourceCompileReq 34-11
 - sourceOptCheck 34-12
 - sourceOptional 34-13
 - sourceReqCheck 34-13
 - sourceRequired 34-14
 - to be added in future 34-17
 - translation into shell commands 24-9
 - unAlias 34-14
 - unProdDir 34-14
 - unsetupEnv 34-15
 - unsetupOptional 34-15
 - unsetupRequired 34-16
 - use with ACTION keyword 34-1
 - writeCompileScript 34-16

G

- g option for user-defined chain 1-5
- groff command
 - ascii output 38-6
 - man option 38-1
 - PostScript output 38-6
- GROUP: keyword 35-3
 - description 27-5
 - use in table files 35-3
 - use in updconfig file 31-2

H

- hardcoded paths problem 15-4
- help on UPS/UPD commands 2-1, 10-1
- help online
 - ups help command 22-41
- html directory 15-7
- HTML_SOURCE_DIR keyword 22-47
 - description 27-5
- HTML_TARGET_DIR keyword 22-47, 30-2
 - description 27-5

I

- include directory 15-7
- independent table file 17-5
- Info directory 15-7
- INFO_SOURCE_DIR keyword 22-47
 - description 27-5
- INFO_TARGET_DIR keyword 22-47, 30-2
 - description 27-5
- init.d directory
 - location 14-1
- initializing UPS environment 1-9
 - courtesy links to files 1-9

- INSTALL_NOTE file 7-1, 15-6, 19-1
 - configuring product 22-15
 - mention of node/flavor-specific functions 10-4
 - mention of unconfigure actions 10-8
 - sample 16-9
- installation methods for UPS products, summary 3-1
- installer accounts
 - choosing 11-1
 - file system semantics 11-2
 - multiple 11-1, 11-2
 - products account 11-1
 - separate by product category 11-2
 - setting gid 11-1, 11-2
 - single 11-1
 - UPD configuration issues 11-2
- installing a product
 - choose whether to declare qualifiers 3-8
 - components to download (using FTP) 7-1
 - configuring 3-9
 - declare manually after FTP download 7-4
 - for development/testing 5-3
 - interruption during install 3-8
 - into AFS space 8-3
 - into private database 11-9
 - KITS product categories 17-3
 - KITS special product registration 17-3
 - local install using AFS UPD 8-2
 - onto distrib node 17-3
 - pass options to local declare 5-2
 - procedural checklist when using UPD 5-3
 - products requiring special privileges 8-1, 12-6
 - root privileges 12-6
 - table file product 17-5
 - tailoring 3-9, 22-67, 22-69
 - troubleshooting 9-1, 10-17
 - ups installasroot command 12-6
 - using FTP 7-1, 7-2, 7-4
 - using UPD 5-1
 - using UPP 6-1
 - with all dependencies (using UPD) 5-5
 - with different name than on server 3-8
 - with no dependencies (using UPD) 5-7
 - with required dependencies (using UPD) 5-7
- instance
 - declare a chain for 10-4
 - definition 1-4
 - determine if update needed 10-13
 - determine instance to act upon 26-1
 - install and declare 5-1
 - specification via chain or version 25-4
 - specify multiple ones in command 25-3
 - verify integrity of 10-10
- instance matching 26-1
 - in chain file 26-3
 - in table file 26-3
 - in updconfig file 31-2
 - in version file 26-3
 - use of flavor and qualifiers 26-4
- instance selection by chain 1-4
- instance specification on command line 25-4
- internal command processes 24-9

K

- K option
 - description for use with ups list 22-46
 - keyword arguments 2-3, 22-46
 - with upd list 4-2
 - with ups depend or upd depend 2-8, 22-30
- keywords 27-1, 28-1
 - case (in)sensitivity of 27-2
 - DECLARED 10-6
 - DECLARER 10-6
 - definition 27-2
 - in ups list output 2-3
 - list with descriptions 22-47, 27-3
 - list with file types 27-3
 - MODIFIED 10-6, 10-13
 - MODIFIER 10-6
 - overriding values 27-3
 - syntax 27-2, 27-8
 - use of @ symbol 22-46
 - used with -K option in ups list 2-3
 - user-defined 27-2
- KITS 4-1
 - adding products to 23-8
 - dbconfig file 21-1
 - FermiTools 7-1, 21-3
 - fnalonly products 21-3
 - product categories 21-3, 23-8
 - product registration for special categories 21-3
 - proprietary products 21-3
 - registration 4-6, 7-2
 - updconfig file 21-2
 - updconfig pre and postdeclare actions 21-4
 - using FTP to download products 7-1
 - US-only products 21-3
- KITS distribution database 17-3

L

- lib directory 15-7
- licensed products
 - permissions 11-3
- link for hard-coded paths 36-2
- links to initialization files 1-9
- list all current products 22-49
- list all fields for a product 2-6, 22-51
- list dependencies on distribution node 4-5, 23-15
- list product dependencies 2-7, 22-27
- list products in database 2-4, 22-49
- list products on distribution node 23-31
 - use in troubleshooting product installs 9-1
- location of database files 11-6
- location of product files, considerations 11-4, 11-5

M

- man directory 15-7
- man -k command 38-3

- man page
 - ascii output 38-6
 - convert to html 38-6
 - determine directory for 11-6
 - file names 38-1
 - groff 38-1
 - information categories 38-3
 - location of files 38-1
 - nroff 38-1
 - nroff output file 38-5
 - nroff source file 16-3, 38-4
 - PostScript output 38-6
 - section numbers 38-1
- MAN_SOURCE_DIR keyword 22-47
 - description 27-5
- MAN_TARGET_DIR keyword 22-47, 30-2
 - description 27-5
- man2html command 38-6
- managing distribution node 20-1
- matching product instance
 - in chain file 26-3
 - in table file 26-3
 - in updconfig file 31-2
 - in version file 26-3
 - use of flavor and qualifiers 26-4
- MODIFIED keyword 10-6, 22-47
 - description 27-5, 28-2, 29-2
 - updating 22-71
 - used to determine if update needed 10-13
- MODIFIER keyword 10-6, 22-47
 - description 27-5, 28-2, 29-2
 - updating 22-71
- multiple databases
 - adding a private database 11-9
 - AFS and local 8-2
 - and your UPD configuration 3-4
 - configuring UPD for 31-9
 - database selection algorithm 26-1
 - default database 1-8
 - how UPD selects a database 5-2, 26-1
 - reasons for using 11-6
 - specifying \$PRODUCTS 1-8, 25-2
 - support for 1-6
 - z option for specifying database 24-5

N

- new chain 1-4
- news directory 15-7
- NEWS_SOURCE_DIR keyword 22-48
 - description 27-6
- NEWS_TARGET_DIR keyword 22-48, 30-2
 - description 27-6
- NFS-mounted database
 - using local database with 12-1
- NIS cluster 12-1
- node.products file 14-3, 14-5
 - permissions 14-4
- notational conventions INT-3
- nroff command 38-4
 - for man page 16-3
 - man option 38-1, 38-5
- NULL flavor 1-3

- number options (-0 through -3) 2-2, 22-36
 - usage information 25-4

O

- old chain 1-4
- online help
 - ups help command 22-41
- option flags
 - command-specific info in reference chapters
 - embedded spaces in arguments 25-2
 - grouping in commands 25-2
 - invalid arguments 25-3
 - multiple arguments 25-2
 - multiple occurrences 25-3
 - wildcards 25-4
- option usage in commands 25-4
- option_list product
 - description 20-12
- order of command line elements 25-1
- ORIGIN keyword 22-48
 - description 27-6, 28-2
- OS determination using ups flavor 2-1, 22-36
- overlaid products 1-6, 16-7, 27-11
- overlays 1-6, 16-7, 27-11

P

- parent product determination 10-8, 22-79
- parse ups list output
 - in perl 22-52
 - in sh script 22-53
- pathAppend function
 - description 34-8
- pathPrepend function
 - description 34-8
- pathRemove function
 - description 34-9
- pathSet function
 - description 34-9
- perl
 - parse ups list output in 22-52
 - version for use with CYGWIN 11-7
- permissions
 - configuring UPD to set for product files 11-2
 - database files 11-3
 - extra security 11-3
 - on downloaded products 3-7
 - on files created in distrib database 20-7
 - unwound tar files 11-2
- pointers in database files 11-6
- pre-built binary products 16-5
 - inserting into template_product 18-4
 - pre-build checklist 19-1
- PROD_DIR keyword 22-48, 27-9
 - as set by -r option 24-4
 - description 27-6, 28-2
- PROD_DIR_PREFIX keyword 3-4, 22-48, 27-9, 30-2
 - description 27-6
- prodDir function
 - description 34-9
- product announcement checklist 19-3

- product categories in KITS 17-3
 - default 21-3
 - FermiTools 21-3
 - FNAL only 21-3
 - proprietary 21-3
 - registration for special categories 17-3
 - U.S. only 21-3
- product dependencies (See dependencies)
- product dependency matching 26-2
- product development 16-7
 - announcement policies for new products 17-10
 - checklist for building product 19-2
 - checklist for distributing to KITS 19-3
 - checklist for pre-build 19-1
 - checklist for product announcements 19-3
 - checklist for testing 19-2
 - code management system 16-6
 - compile script 37-1
 - configure script 36-1
 - configure third-party product 16-6
 - current script 36-3
 - declaring product during development 16-2
 - distributing the product 17-1
 - documentation location 15-7
 - example procedure for simple product 16-1
 - man page creation 16-3
 - overlaid products 16-7
 - pre-build checklist with template_product 19-1
 - pre-built binaries 16-5
 - prep for rebuilding 16-6
 - read-only variables 34-18
 - recommendations
 - fully-specified flavor 15-1
 - location determination 15-2
 - nonuse of /usr/local/bin 15-2
 - nonuse of /usr/local/products 15-3
 - reproducible build procedure 15-3
 - self-containment 15-2
 - shell-independence 15-1
 - system-independence 15-3
 - sample directory hierarchy 16-2
 - selection of build node 16-7
 - simple build procedure 16-1
 - start script 36-3
 - stop script 36-3
 - table files 35-1
 - sample 16-2
 - tailor script 36-3
 - testing product 16-4, 18-5
 - third-party products 15-3
 - uncurrent script 36-3
 - unflavored scripts 16-4
 - using template_product 18-1
 - vendor-supplied products, rebuilding 16-6
- product development tools
 - buildmanager 15-5
 - CVS 15-5
 - template_product 15-6
- product distribution
 - announcement policies for new products 17-10
 - overview 17-1
 - using template_product 18-1, 18-6
 - via CD-ROM 20-14
- product distribution node (See distribution node)
- product documentation 15-7
- product files
 - configure UPD to set location 11-4, 11-5
 - location 11-4, 11-5
 - ownership 11-3
 - permissions 11-3
- product flavor 1-3
- product installation (See installing a product)
- product instance (see instance)
- product instance matching (See instance matching)
- PRODUCT keyword 22-48
 - description 27-6, 28-2, 29-2
- product registration for KITS 21-3
- product removal (See remove a product)
- product root directory 15-6
 - definition 1-3
 - locate using ups list -K 22-52
 - simple example of structure 16-2
- product use statistics 27-9
- product version 1-3
- products account 11-1
- products area 3-4
 - adding a new one 11-9
 - as set in UPD config 3-4
 - choosing location 11-4
 - defining during UPS bootstrap 13-2
 - for development/testing 11-9
 - for KITS 21-1
 - PROD_DIR keyword 27-6, 28-2
 - PROD_DIR_PREFIX keyword 27-6
 - structure of product root directory 15-6
 - unwind product tar files into 7-3
- products for use only at FNAL 21-3
- products for use only in U.S. 21-3
- products requiring build 16-6
 - build script recommendations 15-3
 - inserting into template_product 18-4
 - pre-build checklist 19-1
- proprietary products 21-3
 - on fnkits 4-8

Q

- qualifiers
 - choosing whether to declare them 3-8
 - description 24-8
 - mixing required and optional 24-9
 - optional 24-9
 - overview 1-4
 - required 24-8
 - use in instance matching 26-4
- QUALIFIERS keyword 22-48
 - description 27-6, 28-3, 29-2

R

- reader comment solicitation INT-5
- README file 7-1, 15-6, 19-1
 - sample 16-8
- read-only variables 34-18
 - PRODUCTS 34-19
 - to be added in future 34-21
 - UPS_COMPILE 34-19

- UPS_EXTENDED 34-19
- UPS_OPTIONS 34-19
- UPS_ORIGIN 34-19
- UPS_OS_FLAVOR 34-19
- UPS_PROD_DIR 34-19
- UPS_PROD_FLAVOR 34-19
- UPS_PROD_NAME 34-19
- UPS_PROD_QUALIFIERS 34-19
- UPS_PROD_VERSION 34-19
- UPS_THIS_DB 34-19
- UPS_UPS_DIR 34-20
- UPS_VERBOSE 34-20
- rebuilding product 16-7
- registering products for KITS 21-3
- RELEASE_NOTES file 19-1
 - sample 16-9
- remove a product 10-7, 22-79
 - unconfiguring 10-9
 - using UPP 10-8, 10-10
 - using ups undeclare command 10-8, 22-77
- remove a product component
 - from distrib node 17-8
- remove access to product 2-10, 22-11
- remove product from distrib node 17-8
 - using template_product 18-8
- retrieve file or dir from distribution node 10-15
- retrieve product from distribution node 5-1
- reversible functions 33-2
 - definition 34-1

S

- searchlog.cgi script 20-13
- selecting database for dependency install using UPD 5-3
- selecting database for product install using UPD 5-2
- setup command 1-1, 2-8, 22-5
 - associated environment variables 22-5
 - for chained instance 2-9
 - for current instance 2-9
 - for unchained instance 2-9
 - reference 22-3
 - special options 2-9
 - test if setup would succeed 10-16, 22-33
 - use in troubleshooting problem installations 9-1, 10-17
 - v option for use in troubleshooting 9-1, 10-17
- setupEnv function
 - description 34-10
- setupOptional function
 - description 34-10
 - use to define dependencies 35-4
- setupRequired function
 - description 34-10
 - use to define dependencies 35-4
- setups.[c]sh files 1-9
 - courtesy links to 12-3
 - determine directory for 11-6
 - pointers to 11-6
- SETUPS_DIR keyword 22-48, 30-2
 - description 27-6
- sh
 - parse ups list output in a script 22-53

- shell script products
 - inserting into template_product 18-4
 - pre-build checklist 19-1
- simulate command 9-1, 10-17
- source code
 - revision tracking 17-9
 - storage in CVS 17-9, 18-8
- sourceCompileOpt function
 - description 34-11
- sourceCompileReq function
 - description 34-11
- sourceOptCheck function
 - description 34-12
- sourceOptional function
 - description 34-13
- sourceReqCheck function
 - description 34-13
- sourceRequired function
 - description 34-14
- src directory 15-7
- stage1.sh file 13-1, 13-5
- stage2.sh file 13-5
- stanzas
 - table file 35-1
 - UPD config file 31-1
 - UPP subscription file 6-1, 32-2
- START action 36-3
- start script 14-3, 36-3
- statistics
 - how to gather 11-10, 27-9
 - output 27-10
- STATISTICS keyword 22-48, 30-2
 - as set by -L option 24-3
 - description 27-6, 28-3
 - detailed description of use 27-9
 - output from 27-10
- STOP action 36-3
- stop script 14-3, 36-3
- subscription file for UPP
 - creating 6-1
 - reference 32-1
 - sample for product installation 6-3
- SUFFIX keyword 20-9, 20-10
- syntax of UPS/UPD commands 1-8, 25-1

T

- table files 1-6
 - compile script used with 37-1
 - detailed description 35-1
 - examples
 - action present for some instances only 35-8
 - execute one action or another 35-8
 - grouping 35-6
 - use of FLAVOR=ANY 35-6
 - with user-defined keywords 35-7
 - grouping information in 35-3
 - information storage format 27-2
 - instance matching within 26-3
 - keywords 27-2
 - locate using ups list -K 22-52
 - location specification 28-5
 - naming 35-1

- ordering elements in 35-3
- overwrite 10-14
- read-only variables available for use in 34-18
- recommendations to developers 35-2
- sample for simple product 16-2, 16-4
- stanzas 35-1
- structure and contents 35-2
- test if needs update 10-13
- undoing reversible functions 33-2
- V option for debugging 24-9
- TABLE_DIR keyword 22-48
 - description 27-6, 28-3
- TABLE_FILE keyword 22-48, 27-10
 - description 27-6, 28-3
- tailor a product instance 3-9, 22-69
- TAILOR action 3-9, 22-67, 22-69, 36-3
- tailor script 36-3
- tar file creation
 - by upd addproduct 17-1, 23-7
 - using template_product 18-5
- template_product 15-6, 17-2
 - adding build instructions 18-4
 - to top-level Makefile 18-4
 - checklist for building product 19-2
 - checklist for distributing to KITS 19-3
 - checklist for pre-build 19-1
 - cloning 18-2
 - customizing product tar file 18-5
 - downloading 18-2
 - editing top-level Makefile 18-3
 - inserting pre-built binaries 18-4
 - inserting product requiring build 18-4
 - inserting shell scripts 18-4
 - inserting your product 18-4
 - Makefile (top-level) 18-3
 - overview 18-1
 - removing product from distrib node 18-8
 - running a build procedure 18-4
- temporary script
 - prevent deletion 24-9
- test chain 1-4
- test directory 15-7
- testing products 18-5
 - checklist 19-2
- third-party products 15-3
- toInfo directory 15-6
- toman directory 15-6

U

- umask 3-7
- unAlias function
 - description 34-14
- unchain
 - as action in table files 33-3
 - replace chain on distrib node using upd modproduct 17-7
 - use ups undeclare to remove chain 10-6, 22-77
- UNCONFIGURE action 10-9, 22-75, 36-1
- unconfigure script 36-1
- UNCURRENT action 36-3
- uncurrent script 36-3
- undeclare a chain 10-6, 22-77
- undeclare a product instance 10-7, 22-79
 - using UPP 10-8
 - using ups undeclare command 10-8
- undoing chains in table files 33-3
- unflavored scripts 16-4
- UNIX Product Distribution
 - overview 1-1
- UNIX Product Poll 32-1
 - overview 1-1
- UNIX Product Support
 - overview 1-1
- unknown command handler
 - description 33-3
- unProdDir function
 - description 34-14
- unsetup command 2-10, 22-11
 - \$SETUP_UPS variable 1-10
 - behavior with dependencies 22-11
 - reference 22-9
 - use of \$SETUP_<PRODUCT> variable 22-6, 22-11
- unsetupEnv function
 - description 34-15
- unsetupOptional function
 - description 34-15
- unsetupRequired function
 - description 34-16
- UNWIND_ARCHIVE_FILE keyword 20-9, 20-10
 - description 27-6
 - use in updconfig 31-4
- UNWIND_PROD_DIR keyword 3-4
 - description 27-7
 - use in updconfig 31-3
- UNWIND_TABLE_DIR keyword
 - description 27-7
 - use in updconfig 31-4
- UNWIND_UPS_DIR keyword
 - description 27-7
 - use in updconfig 31-3
- UPD
 - command syntax 1-8
 - configuration file
 - info for installers 3-3
 - overriding default 3-4
 - reference 31-1
 - overview 1-1
 - procedural checklist for installation 5-3
- upd addproduct command
 - adding table file product 17-5
 - adding typical product 17-3
 - chains 17-3, 23-7
 - detailed functions 20-2
 - internal processes 23-8
 - reference 23-3
 - response of distrib node 20-2
 - tar file creation 17-1, 23-7
- upd cloneproduct command
 - reference 23-11
- UPD commands
 - defaults 1-8
 - dependency matching 26-2
 - instance matching 26-1
 - interaction with distrib node 20-1
 - option flag grouping 25-2
 - option usage 25-4
 - order of command line elements 25-1

- specifying version/chain 25-1
- specifying multiple products 25-3
- UPD configuration file 27-3
 - AFS issues 8-2
 - distrib node 20-9
 - KITS database pre and postdeclare actions 21-4
 - pre and postdeclare actions 20-10
 - examples 31-7
 - AFS 31-10
 - distrib node config 31-10
 - distribution from fnkits 31-8
 - multiple dbs and distrib nodes 31-9
 - for KITS database 21-2
 - info for installers 3-3
 - organization 31-1
 - overriding default 3-4, 31-1
 - overview 27-1
 - pre and postdeclare actions 31-5
 - product matching 31-2
 - reference 31-1
 - required location definitions 31-3
 - sample location definitions 31-5
 - setting file permissions 11-3
 - stanzas 31-1
- upd delproduct command 17-8
 - reference 23-13
- upd depend command 4-5
 - reference 23-15
- upd exist command 10-16
 - reference 23-17
- upd fetch command 10-15
 - reference 23-19
- upd get command
 - reference 23-23
- upd install command 5-1
 - database selection 5-2
 - database selection for dependencies 5-3
 - detailed functions 20-2
 - G (pass options to local declare) 5-2, 23-28
 - internal processes 23-29
 - procedural checklist for installation 5-3
 - reference 23-25
 - response of distrib node 20-2
 - summary of functions it performs 3-1
 - syntax and commonly used options 5-1, 23-25
 - use to determine if product update needed 10-13
- upd list command 4-1
 - reference 23-31
- upd modproduct command 17-6, 17-7
 - reference 23-33
 - response of distrib node 20-2
- upd move_archive_file script 20-2
- upd moved_ups_dir script 20-2
- _UPD_OVERLAY keyword 16-7, 27-11
 - description 27-8
- upd reproduct command
 - reference 23-39
- upd update command 10-13, 10-14
 - reference 23-41
- upd verify command
 - reference 23-45
- upd.cgi script 20-2, 20-11
 - access restrictions 20-6
 - description 20-5
- UPD_USERCODE_DB keyword 22-48
 - description 27-7
- UPD_USERCODE_DIR keyword 3-4, 22-48, 30-2
 - description 27-7
 - on fnkits 21-2
- update product
 - determine if update needed 10-13
 - using UPD 10-13
 - using UPP 10-13
- updconfig file (see UPD configuration file)
- updconfig.template file 31-1, 31-7
- updusr.pm file 31-1
- upgrading UPS installation 11-8
- UPP
 - automate upp command via cron 6-4
 - command syntax 6-4
 - monitor products on distribution node 4-3
 - notification of update needed 4-3, 10-13
 - overview 1-1
 - remove a product 10-8, 10-10
 - subscription file
 - creating 6-1
 - definition 4-3
 - sample for product installation 6-3
 - uses 32-1
- upp command 4-3, 6-1
 - automation via cron 6-4
 - reference 23-47
 - syntax 4-4, 6-4
- UPP subscription file
 - adding instructions 32-2
 - available functions 32-3
 - creating 6-1
 - definition 4-3
 - header description 32-1
 - instance matching 32-2
 - reference 32-1
 - sample 32-3
 - sample for product installation 6-3
 - stanza description 32-2
- UPS
 - aliases defined 1-10
 - benefits of methodology 1-2
 - chains 1-4
 - command syntax and defaults 1-8
 - database 1-1
 - database directory specification 1-10
 - motivation for methodology 1-2
 - multiple database support 1-1
 - multiple product flavor support 1-3
 - multiple product version support 1-2, 1-3
 - overview 1-1
 - pointer to product root directory (\$UPS_DIR) 1-10
 - product instance 1-4
 - product version 1-3
 - products distributed and managed by 1-3
 - upgrading your UPS installation 11-8
 - use without a database 1-7, 11-7
- UPS commands
 - "-?" for usage information 2-1
 - "uncommands" as action keyword values 34-1
 - as ACTION keyword 33-1
 - database selection 26-1
 - defaults 1-8
 - dependency matching 26-2

- instance matching 26-1
- keeping statistics on 11-10, 27-9
- option flag grouping 25-2
- option usage 25-4
- order of command line elements 25-1
- specifying multiple products 25-3
- specifying version/chain 25-1
- UPS configuration file 27-3
 - defining directory locations in 11-6
 - for KITS database 21-1
 - for local database on fnkits 21-1
 - keywords used in 30-1
 - overview 27-1
 - reference 30-1
 - sample 30-2
- ups configure command 3-9
 - reference 22-13
- ups copy command 22-19
 - reference 22-17
- UPS database
 - \$PRODUCTS variable 1-10
 - \$UPS_EXTRA_DIR variable for AFS 12-5
 - .upfiles subdirectory 1-6
 - .upsfiles subdirectory 1-6
 - checklist for creating a database 11-9
 - choosing single or multiple 11-6
 - configuring local to work with AFS 12-2
 - create a private database 11-9
 - create local database to work with AFS 12-2
 - declare a product instance to 3-5, 10-1
 - declaring products into local (not AFS) 12-4
 - definition 1-6
 - for development/testing 11-9
 - installing products into local (not AFS) 12-5
 - list all current products in 2-4
 - list product information 2-2
 - listed in upsd_b_list file 12-2
 - multiple (See multiple databases)
 - NFS mounted 12-1
 - permissions for files (distrib node) 20-7
 - providing access to multiple databases 12-2
 - setting up your own 5-3
 - with AFS 12-2
 - standard naming conventions for use with AFS 12-2
 - structure and contents 1-6
 - using AFS and local 5-3
 - using UPS without a database 1-7, 11-7
- UPS database files 1-6
 - chain files 1-6, 29-1
 - check for inconsistencies 10-10
 - editing 10-11
 - keywords 27-1
 - overview 27-1
 - UPD configuration file 31-1
 - UPS configuration file 30-1
 - version files 1-6, 28-1
- ups declare command 3-6, 10-3
 - as used internally by upd install 5-2
 - reference 22-21
 - specifying database 3-5, 10-2
 - specifying table file path 3-5, 10-2
 - specifying ups directory 3-5, 10-2
 - syntax and common options
 - for declaring chain 10-4
 - for declaring instance 3-5, 7-4, 10-2
 - use during development 16-2
 - use to declare chain 10-4
 - use to declare instance 3-5, 10-1
- ups depend command 2-7, 10-8, 22-79
 - reference 22-27
- ups directory 3-5, 7-1, 10-2, 15-6, 27-11
 - description 15-6
 - locate using ups list -K 22-52
 - overwrite 10-14
 - test if needs update 10-13
- UPS environment (See environment)
- ups exist command 10-16, 22-33
 - reference 22-31
- ups flavor command 2-1
 - H option (specifies other flavor) 22-36
 - l option (returns flavor table) 22-36
 - number options (specify OS level) 2-2
 - obtain flavor levels 2-2
 - obtain flavor table 2-2
 - reference 22-35
- ups get command
 - reference 22-39
- ups help command
 - reference 22-41
- UPS initialization file 11-6
- ups installasroot command 12-6
- ups list command 2-2, 3-6, 10-3, 10-5
 - condensed output 2-3, 22-46
 - default output fields 22-45
 - for db managers and product installers 27-1
 - formatted output 2-3, 22-45
 - K option
 - for script-readable format 2-3, 22-46
 - keyword arguments 22-46
 - use to locate product files 22-52
 - keywords for -K option 2-3
 - list all current products 2-4
 - list all output fields 2-6
 - long listing 22-51
 - parse output
 - in perl 22-52
 - in sh script 22-53
 - reference 22-43
- ups modify command
 - editing database files 10-11
 - reference 22-55
- UPS product overlay (See overlays)
- UPS product requirements (See dependencies)
- UPS products
 - accessibility 10-16
 - announcement policies 17-10
 - bin directory 15-6
 - build and distribute using template_product 18-1
 - catman directory 15-7
 - compilation options 1-4
 - definition 1-3
 - directory structure 15-6
 - distribution restrictions 20-11
 - distribution via CD-ROM 20-14
 - doc directory 15-7
 - documentation storage 15-7
 - examples directory 15-7
 - files and directories to include 19-1
 - hardcoded locations 15-3
 - html directory 15-7

- include directory 15-7
- Info directory 15-7
- INSTALL_NOTE file 15-6
- installation methods, summary 3-1
- installed with different name than on server 3-8
- interruption during installation 3-8
- lib directory 15-7
- list on distribution node 4-1
- man directory 15-7
- news directory 15-7
- overlays 16-7
- permissions set at installation 3-7
- proprietary products
 - on fnkits 4-8
- qualifiers 1-4
- README file 15-6
- special categories, flagging 20-12
- src directory 15-7
- support levels 17-10
- test directory 15-7
- third-party 15-3
- toInfo directory 15-6
- toman directory 15-6
- ups directory 7-1, 15-6, 27-11
- ups script 14-1
- ups setup command (for troubleshooting) 9-1, 10-17
- ups start command 14-2, 14-5
 - reference 22-59
 - usage in autostart 14-3
- ups stop command 14-2
 - reference 22-63
 - usage in autostart 14-4
- ups tailor command 3-9, 22-69
 - reference 22-67
- ups touch command
 - reference 22-71
- ups unconfigure command 10-7, 10-9, 22-79
 - reference 22-73
- ups undeclare command
 - reference 22-77
 - remove chain 10-6, 22-77
 - remove product instance 10-7, 10-8, 22-79
 - syntax and common options
 - for chain removal 10-6
 - for product removal 10-8
 - y and -Y options to remove root directory 10-8
- ups verify command 10-10
 - reference 22-81
 - run by ups modify 10-11
 - use in troubleshooting problem installations 9-1, 10-17
- ups.cgi script 20-2
 - description 20-5
- UPS/UPD/UPP installation components 1-1
- UPS_ARCHIVE_FILE keyword 20-9, 20-10
 - description 27-7
- UPS_ARCHIVE_FILES keyword
 - use in updconfig 31-4
- UPS_DB_VERSION keyword 30-2
 - description 27-7, 28-3, 29-2
- UPS_DIR keyword 22-48, 27-11
 - as set by -U option 24-5
 - description 27-7, 28-3
- UPS_EXTENDED variable 24-7

- UPS_PROD_DIR keyword 3-4
 - description 27-7
 - use in updconfig 31-3
- ups_shutdown script 14-1, 14-2, 14-5
- ups_startup script 14-1, 14-2, 14-5
- UPS_TABLE_DIR keyword
 - description 27-7
 - use in updconfig 31-3
- UPS_TABLE_FILE keyword
 - description 27-8
 - use in updconfig 31-4
- UPS_THIS_DB keyword
 - description 27-7
 - use in updconfig 31-3
- UPS_UPS_DIR keyword
 - description 27-8
 - use in updconfig 31-3
- upsdb_list file 12-2
 - for AFS 12-5
- upsdb_list variable 13-3
- ups-decl.cgi script 20-2, 20-11
 - access restrictions 20-6
 - description 20-5
- user comment solicitation INT-5
- USER keyword
 - description 27-8
- user-defined chains 1-4
- user-defined commands 33-3
- user-defined keywords 27-2
- US-only products 21-3

V

- variables (read-only) defined within UPS 34-18
- vendor-supplied products
 - rebuilding 16-6
- verbose command output (-v) 9-1, 10-17
- version files 1-6, 22-79
 - and product removal 10-7
 - creating 28-1
 - description 28-1
 - examples 28-3
 - information included in 28-1
 - information storage format 28-1
 - instance matching within 26-3
 - location 28-1
 - overview 27-1
 - table location specification in 28-5
- VERSION keyword 22-48
 - description 27-8, 28-3, 29-2
- version of product 1-3
- version specification in commands 25-1

W

- web server
 - access file 20-11
 - log file on fnkits 21-7
 - on distrib node 20-1
 - prerequisites for cgi scripts 20-7

writeCompileScript function
description 34-16
www
download products from 16-5