

# Chapter 3: 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 23: *UPS Command Reference* for command descriptions and command-specific option descriptions, and Chapter 26: *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**.

## 3.1 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 23.8 *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
```

```
Linux+2.4.18-5
```

```
Linux+2.4.18
```

```
Linux+2.4
```

```
Linux+2
```

```
Linux
```

```
NULL
```

```
ANY
```

You can specify a particular level using the number options: **-0** (NULL), **-1** (basic OS only), **-2** (basic OS + version), **-3** (basic OS + version + release), **-4** (basic OS + version + release + patch). No number option gives you basic OS + version + release + patch + build. For example:

```
% ups flavor
```

```
Linux+2.4.18-5
```

```
% ups flavor -4
```

```
Linux+2.4.18
```

```
% ups flavor -3
```

```
Linux+2.4
```

```
% ups flavor -2
```

```
Linux+2
```

```
% ups flavor -1
```

```
Linux
```

```
% ups flavor -0
```

```
NULL
```

## 3.2 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 23.11 *ups list*.

### 3.2.1 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.

### 3.2.2 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 23.11 *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 <b>UPS</b> 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 23.11.4 *More Detailed Description* under the reference section 23.11 *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 29.4 *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

### 3.2.3 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(mimertools)
    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.

## 3.3 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 23.6 *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
|__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
| | |__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
| | | |__imagelibs v1_0 -f SunOS+5 -z /afs/fnal.gov/ups/db
| | | | |__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 3.2.2 *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"
"mimetools" "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
|__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
|__ispell v3_1b -f SunOS+5 -z /afs/fnal.gov/ups/db -g
current
```

## 3.4 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 23.1 *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 unsetup first.

## 3.4.1 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 3.2 *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 23.1 *setup*.

## 3.4.2 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 23.1 *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. A qualifier may be preceded by one of two operators (**-q [+|?]*qualifier***) which explicitly makes the qualifier *mandatory* (+) or *optional* (?). A qualifier not preceded by either operator is *variable*. Variable is the most commonly used form, and means that the qualifier is:

- mandatory when used with a command that sets a qualifier, e.g., **ups declare**, **upd addproduct**
- optional when used with a command that must find a product instance (with or without the qualifier declared) and then do something with the product instance, e.g., **setup**, **ups list**, **ups depend**, **upd install**, and so on

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 (or no operator)<sup>1</sup>, e.g.,

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

This will give you the second listed instance above. But depending on the available instances in your case, the matched instance may or may not have the qualifier.

---

1. The "?" is no longer strictly necessary here, you'd get the same result without it. The "?" is supported for backwards compatibility.

See the reference section 25.2.4 *-q* for more information on specifying qualifiers on the command line.

## 3.5 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 23.2 *unsetup* for an explanation of how the behavior of **unsetup** depends on the product's implementation.