

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 35.3 *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 35.3 *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 28.4 *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 24: *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 23: *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*.