

Chapter 18: Making Products Available For Distribution

This chapter describes the processes of adding, updating, deleting and “cloning” product instances or components¹ on a product distribution system. Information on creating tar files, using Fermilab CVS repositories and announcing products is also provided.

18.1 Product Distribution Overview

A set of **UPD** commands has been developed for adding, updating and deleting products on a distribution node. They use the central Fermilab product distribution node *fnkits.fnal.gov* as the default distribution node, and declare products in the `KITS` database. The commands can be used to distribute products to any properly configured product server.

These **UPD** commands include:

upd addproduct	adds a product instance to a product distribution database
upd cloneproduct	creates a new product instance on a distribution node by copying one that is already there and changing one or more of its identifying elements
upd delproduct	deletes a product declaration from a distribution database; it also removes any associated tar file, table file and/or <code>ups</code> directory
upd modproduct	modifies a product instance that already exists in a distribution database; it allows you to replace a table file or <code>ups</code> directory, or to add or change chain information for the product
upd reproduct	is equivalent to a upd delproduct followed by a upd addproduct ; it can be used only when the replacement product instance has the same set of identifiers as the one destined for removal

1. “Components” are defined as table files and `ups` directories.

These commands are fully described in Chapter 24: *UPD/UPP Command Reference*.

Before preparing to distribute a product, you should verify that it is complete, tested, and **UPS**-compliant. It is optional to create a tar file of your product prior to running `upd addproduct`, as discussed in section 18.2 *Creating Product Tar Files*. Keep in mind that the **UPD** configuration on your target distribution node determines the locations in which products get installed and declared on that node, and where their auxiliary files/directories get stored. The configuration on the distribution node may bear no resemblance to that of your local development system, or to that of an end user node. Once your product has been added to a distribution node, you need to make the appropriate announcements regarding product availability (see section 18.10 *Product Announcement Policies*).

18.2 Creating Product Tar Files

You can choose whether to make your own tar file before adding your product to a distribution node or to let **UPD** make it for you. The advantage of making it yourself is that you have control over its contents.



Note that it is not necessary that the product instance's table file or `ups` directory be included in the product root directory or, consequently, in the tar file. Some products may not have one, the other, or both of these components. On the other hand, other products (e.g., bundled products) may consist only of a table file, in which case no tar file is needed. If these components exist and are located outside of the product root directory, their location must be specified in one of two ways when adding the product to the distribution node:

- on the `upd addproduct` command line
- in a **UPS** database declaration on your development machine (database must be listed in `$PRODUCTS`)

When creating a tar file of a product using the `tar` command, perform the operation from the product root directory. This allows you to use simple relative path names to specify the files to include in the tar file. Use an absolute pathname (preferably to a temporary directory) to specify where to put the tar file. Do *not* use absolute pathnames to specify the files to include in the tar file.



Do *not* use the product root directory as the destination for the created tar file; it causes the tar file to try to include itself and to grow infinitely large.

The following steps illustrate the conventions for packing up a tar file for a **UPS** product called **fred** in such a way that (a) the tar file contains a relative path to the product root directory, and (b) the tar file is put in an appropriate temporary directory:

```
% setup [-d] fred
% cd $FRED_DIR
% tar cvf /tmp/fred_IRIX+5_v1_0.tar .
```

This creates a tar file called `/tmp/fred_IRIX+5_v1_0.tar` with all pathnames relative to the current directory (`$FRED_DIR`).



You should *not* replace the trailing dot in the example above with `$FRED_DIR` because that would force the tar file to contain an absolute path to the `$FRED_DIR` as set on *your* system, instead of a relative path to the `$FRED_DIR` on the *target* system where the tar file will be unwound.



Using **template_product** (described in Chapter 19: *Using template_product to Build and Distribute UPS Products*) allows you to customize the contents of your tar file. See section 19.8 *Customizing your Tar File*.

18.3 Adding a Product

Use the **upd addproduct** command to add a new product instance to a distribution server. If no host name is specified with the **-h** option, **UPD** uses the `fnkits.fnal.gov` host as the default. The required command line arguments differ depending on what components the product has and whether it's been declared to a local **UPS** database and/or archived with tar. Refer to the reference section 24.1 *upd addproduct* for the full command syntax and options for these different situations.

A few notes:



- When using **upd addproduct -h <host>**, use the full hostname (i.e., `hostname.fnal.gov` rather than just `hostname`) to prevent problems when people download the product to off-site user nodes.
- The **-P** option is available to prevent **UPS** from searching in a local database for the product instance. If you use it, you must specify sufficient information on the command line so that **UPS/UPD** can identify and locate all the product components.
- 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.
- If the product is not declared to a local database, you must include **-m <tableFileName>** on the command line. You must also include **-M <tableFileDir>** if the table file is not in the current directory.

18.3.1 Product Categories Defined for KITS

The central Fermilab Computing Division product distribution database KITS, located on the server *fnkits.fnal.gov*, recognizes several different categories of product:

default	regular products added to the KITS 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
fnalonly	products accessible only to the <i>fnal.gov</i> domain
usonly	US-only (United States only) products are accessible only to U.S. government (<i>.gov</i>) and military (<i>.mil</i>) 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., do not use `-O "<options>"`) with `upd addproduct`; your product will automatically be configured to handle the special requirements according to your selection on the form.

18.3.2 Examples

Example 1

We have a product instance with a table file and a `ups` directory (in addition to all the product files) under the product root directory. The table file is in the `ups` directory. The product (we'll call it **foo** version `v1_0`), was developed for the flavor SunOS+5. The tar file has not been made ahead of time. In order for **UPD** to make the tar file for us, the product instance must be declared to a local **UPS** database listed in `$PRODUCTS`.

To add the product to KITS, the command can be entered from a SunOS+5.x machine as:

1. See the **Product Distribution Platform Registration Request** form at http://www.fnal.gov/cd/forms/upd_registration.html

```
% upd addproduct foo v1_0 -2
```

Notice we've used the option `-2` which is equivalent in this case to `-f SunOS+5`. All of the other necessary information gets picked up from the local **UPS** declaration.

If we choose to ignore the local declaration via the `-P` option, we must supply the necessary information in the command:

```
% upd addproduct foo v1_0 -2 -P -r /path/to/prod/root/dir \  
-m v1_0.table -M /path/to/prod/root/dir/ups \  
-U /path/to/prod/root/dir/ups
```

Example 2

Let's use the same product as in Example 1, but assume that a tar file already exists. The pre-made tar file includes the entire structure under the product root directory. The tar file is located on our local machine in `/tmp/foo_v1_0_SunOS+5.tar`. We want to add it to *fnkits* and declare it to the `KITS` database with the full development machine flavor specification, no qualifiers, and no chain.

Assuming this product instance was declared to a local **UPS** database before the tar file was created, we use the command:

```
% upd addproduct foo v1_0 -2 -T /tmp/foo_v1_0_SunOS+5.tar
```

UPD can determine where to find the table file and `ups` directory on the local node by querying the local **UPS** declaration. However, if the product instance had *not* been declared to any local **UPS** database, we would need to specify the table file name and location. We would also need to specify the `ups` directory if it were other than `${UPS_PROD_DIR}/ups`¹, which is the default location. A sample command that would work for this case is:

```
% upd addproduct foo v1_0 -2 -T /tmp/foo_v1_0_SunOS+5.tar \  
-m foo.table -M ups
```

If the command succeeds, **UPD** returns a message indicating that the product was successfully transferred and declared. After the product is added, we can run the `upd list -a` command to see the declaration in `KITS`:

```
% upd list -a foo v1_0  
  
DATABASE=/ftp/upsdbusr  
Product=foo Version=v1_0 Flavor=SunOS+5  
Qualifiers="" Chain=""
```

If we had wanted to declare the product in `KITS` for several flavors (assuming flavor-independence in the product), we could have specified them in the command as follows:

1. `${UPS_PROD_DIR}` is one of a set of local **UPS** read-only variables listed in section 35.6 *Local Read-Only Variables Available to Functions*. It takes the same value as `$(PRODUCT)_DIR`, the product root directory.

```
% upd addproduct foo v1_0 -f IRIX+5:SunOS+5:OSF1_v3 \  
-T /tmp/foo_v1_0_ANY.tar -m foo.table -M ups
```

Example 3

This next product, **footwo** v1_0, has no table file (thus no **-m** or **-M** needed), and it has a **ups** directory external to the product root directory. We want to declare it to the (fictional) node *dist_node.fnal.gov* with the test chain (**-t**), the flavor NULL (**-0**), and the qualifier “debug” (**-q "debug"**):

```
% upd addproduct footwo v1_0 -t0q "debug" \  
-h dist_node.fnal.gov -T /tmp/footwo_v1_0_NULL.tar \  
-U /local/path/to/ups/dir
```

After it is added, we can run the **upd list -a** command to see the declaration on the distribution node:

```
% upd list -a -h dist_node footwo v1_0  
  
DATABASE=/path/to/dist_db  
Product=footwo Version=v1_0 Flavor=NULL  
Qualifiers="debug" Chain="test"
```

18.4 Adding an Independent Table File

You need to use **upd addproduct** to add a new table file product (i.e., a table file that isn't a component of a product instance). Bundled products are usually table files, for example. To replace a table file that is a component of a product instance already declared to the database on the distribution node, use **upd modproduct** as described in section 18.5 *Replacing a Component (Table File or ups Directory)*.

If the independent table file is declared to a local database, the command syntax is:

```
% upd addproduct [<flavor_option>] [<other_options>]  
<product>\ <version>
```

If the table file is not declared, the command syntax becomes:

```
% upd addproduct [-P] <flavor_option> -m  
<tableFileName> \ [-M <tableFileDir>]  
[<other_options>] <product> <version>
```

Example

The product **foothree** v1_0 consists only of a table file (it may be a bundled product); therefore no tar file needs to be specified (no **-T** option). We want to add it and declare it to **KITS** with no chain, no qualifiers, and the flavor **IRIX**. We do not assume that it's been declared to a local **UPS** database:

```
% upd addproduct foothree v1_0 -f IRIX -m foothree.table -M \  
  /local/path/to/table/file
```

The system returns a message saying there is no product root directory. This is correct behavior, and is expected.

After the table file product is added, we can run the **upd list -a** command to see its declaration in **KITS**:

```
% upd list -a foothree v1_0  
  
  DATABASE=/ftp/upsdbusr  
      Product=foothree  Version=v1_0  Flavor=IRIX  
      Qualifiers=""     Chain=""
```

18.5 Replacing a Component (Table File or ups Directory)

Use **upd modproduct** to update the table file or **ups** directory of a product already existing on the distribution node. This command cannot query the local **UPS** database to find information the way **upd addproduct** can; all necessary information must be specified on the command line. To replace a table file, the command syntax is:

```
% 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, the syntax is:

```
% upd modproduct <flavor_option> -U <upsDir> \  
  [-m <tableFileName>] [-M <tableFileDir>] [<other_options>]\  
  <product> <version>
```



If the **ups** directory contains a newer table file that should replace the old one on the distribution node, include the **-m** and **-M** options in the command.

Example: Table File

Let's replace the table file in `KITS` for the product `foo v1_0`, from Example 1 of section 18.3. The new table file, `foo.table`, has replaced the old one in the product instance's local `ups` directory. It doesn't matter if the tar file has been remade, since we're not going to send it anyway.

```
% upd modproduct foo v1_0 -2 -m foo.table \  
  -M /local/path/to/ups/dir
```

If you issue the command from the directory specified by `-M`, then you don't need to include it on the command line.

Example: ups Directory

To replace a product instance's `ups` directory, use the `upd modproduct` command with the `-U` option. Specify as much product instance information on the command line as necessary to uniquely identify the instance in the distribution database to which this directory is to belong. Do not make a tar file of the `ups` directory on your local machine. We illustrate with a product called `foofour v1_0`, flavor `SunOS`, no qualifiers, and use `KITS`.

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, for example:

```
% upd modproduct foofour v1_0 -f SunOS \  
  -U /local/path/to/ups/dir
```

18.6 Adding/Changing a Chain

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 chains. The command syntax is:

```
% upd modproduct <flavor_option> <chain_option> \  
  [<other_options>] <product> <version>
```

1. A product instance can have multiple chains if they are declared together in the same command (e.g., `upd modproduct -g test:current ...`).

Example 1

Product **foo** (of Example 1 in section 18.3) has no chain in its `KITS` declaration. We now wish to declare a test chain for it. We run the `upd modproduct` command with the `-t` option (or `-g test` works too), as follows:

```
% upd modproduct foo v1_0 -f SunOS+5 -t
```

Running `upd list -a` now displays:

```
% upd list -a foo v1_0
```

```
    DATABASE=/ftp/upsdbusr
    Product=foo Version=v1_0 Flavor=SunOS+5
    Qualifiers="" Chain="test"
```

Example 2

This time we want to change an existing chain. Let's change the test chain for **foo** (declared in Example 1, above) to current. This will remove the test chain.

```
% upd modproduct foo v1_0 -f SunOS+5 -c
```

Running `upd list` now displays:

```
% upd list foo v1_0
```

```
    DATABASE=/ftp/upsdbusr
    Product=foo Version=v1_0 Flavor=SunOS+5
    Qualifiers="" Chain="current"
```

Notice that since we were looking for a current version, we didn't need to specify `-a` in the `upd list` command.

Example 3

To remove a chain on an instance without assigning a new one or assigning the chain to a different instance, you can use:

```
% upd modproduct foo v1_0 -f SunOS+5 -g :
```

This often generates warnings, but it works and causes no database problems.

18.7 Deleting a Product or Component

The `upd delproduct` command lets you delete a product declaration plus the product itself and its associated files and directories. The product subdirectory itself does not get deleted. You do not have the choice of leaving an undeclared product in the products area on the distribution node. The command syntax is:

```
% upd delproduct -f <flavor_option> [<other_options>] \  
  <product> <version>
```

Example 1

Let's delete the product `foo v1_0` (from Example 2 in section 18.6):

```
% upd delproduct foo v1_0 -cf SunOS+5
```

Example 2

Let's delete the product `foothree v1_0` from section 18.4. It's just a table file.¹

```
% upd delproduct foothree v1_0 -f IRIX
```

18.8 Cloning a Product

Use `upd cloneproduct` to create a new product instance on a distribution node by copying one that is already there and changing one or more of its identifying elements.

1. If there were a product that consisted only of a `ups` directory (unlikely), `upd delproduct` would work for that too.

The command syntax is:

```
% upd cloneproduct <flavor_option> [<source_options>] \  
  <product> [<version>] -G "<target_options>"
```

where *source* refers to the original instance, and *target* to the cloned one.

To clone a product, you specify the usual **UPS/UPD** options to identify the product, and then use the **-G** option to specify which attributes of the clone should be different from the original.

Why would you want to do this? For example, say that an existing product for the flavor IRIX+5 is found to be appropriate for IRIX+6, too. In this case, you might want the product to appear on the distribution server listed under both flavors. You could install the product on your local system, redeclare it, and add it back to the distribution server, but a much quicker and more efficient way is to use **upd cloneproduct** to *clone* the product instance right on the distribution server. Here is a sample command for doing this:

```
% upd cloneproduct myproduct v1_0 -f IRIX+5 -G "-f IRIX+6"
```

You can put all sorts of options in the **-G** quoted argument list, including product and version (with caveats); so you can even use **upd cloneproduct** to make a clone with a different name, provided the product's table file doesn't specify the product name. For example, to make a clone of **myprod** called **newprod** in **KITS**, you'd issue a command like this:

```
% upd cloneproduct myproduct v1_0 -f IRIX+5 -G "newprod"
```

A few caveats:

- Within the **-G** option structure on the **upd cloneproduct** command line, only include options such that a stanza of the source product's table file can be matched. A failure to match sometimes creates a database inconsistency on the distribution node. In particular, be careful about including qualifiers, e.g., **-G "-q <qualifierList>"**, if there is no stanza for `Qualifiers = qualifierList`.
- If you want product instance clones, one without qualifiers and the other with, add the first instance without qualifiers, and clone it to a new instance with qualifiers. Going the other way is error-prone.
- You can only make a clone with a different product name if the source product's table file doesn't specify the product name.



18.9 Including Source in one of Fermilab's CVS Repositories

Different groups at Fermilab often depend upon each other's software, and people need to be able to rebuild products on occasion. The CVS Product Repositories have been created to provide a structure allowing access to source code with revision tracking. The product eligibility standards are described in the document *Using Fermilab CVS Product Source Repositories*, on-line at http://www.fnal.gov/docs/products/template_product/FermiRepository/FermiRepository.html.

18.10 Product Announcement Policies

The separate groups within the Computing Division have differing policies for informing the group members and the user community about product availability. Here we present a checklist of the kinds of things you will be expected to do when you're ready to make a product available. We refer you to your group leader for information specific to your group.

Events which require notification actions on your part are:

- initially placing a product on *fnkits*, declared as "test" (recommended)
- declaring the product as current on *fnkits*
- installing the product in AFS space
- upgrading the product
- modifying or removing the product on/from *fnkits*

The general types of required actions are:

- Inform your group leader.
- Announce product according to group's policy (newsgroups, product user mailing lists)
- Send email to helpdesk@fnal.gov to inform them about the new product or version. Include information on the kinds of questions to expect, if possible, and where to direct users for help.
- Install the product on *fnalu* for the general Fermilab community, if appropriate.
- Check all the chains on *fnkits* (and *fnalu*) to make sure that older versions, flavors, etc. are no longer chained to current.
- Include source code for eligible product in a CVS Repository.

- Make documentation available on-line under http://www.fnal.gov/docs/products/<product_name>. Include html documentation.
- Fill out the on-line **Computing Division Product Input Form** at <http://cddocs.fnal.gov/cfdocs/productsDB/productinput.html> to inform the products database maintainer about your product arriving on *fnkits*.

