

Chapter 7: Installing Products Using UPP

UPP can be used for several functions as described briefly in section 2.1 *Introduction to UPS, UPD and UPP*, and in detail in Chapter 33: *The UPP Subscription File*. This chapter describes how to use **UPP** to install products.

7.1 Overview of Using UPP to Install Products

UPP requires what we call a *subscription file* which tells it what products to look for on a designated distribution node, and what functions to perform when it detects that new versions of these products are released there. One of the functions **UPP** can perform is product installation. **UPP** does this by running **upd install** (described in Chapter 6: *Installing Products Using UPD*). You can also instruct **UPP** to run **ups declare** commands to resolve dependencies as necessary when a product installation finishes.

Your job is to create a **UPP** subscription file and run the **upp** command. To automate **UPP**'s operations, the **upp** command can be run periodically (for example from **cron**).

7.2 Creating a UPP Subscription File

A subscription file consists of a header followed by at least one stanza. The header includes an email address for notification, the distribution node to query, and other “administrative” information. Each stanza has three parts:

- identification of a product or particular instances of a product
- identification of the condition(s) for which you want **UPP** to perform the instructions you give it
- a list of instructions, or functions to perform, for each condition

A stanza is bracketed by the lines `begin` and `end`. The number of stanzas per file is not limited. A stanza cannot refer to multiple products, however there can be multiple stanzas for the same product (e.g., for treating different instances of the same product differently).

7.2.1 Create the Header

The header should look similar to this example (explanations are on the right):

<code>file = upp</code>	This identifies the file as a UPP subscription file.
<code>mail_address = joe@fnal.gov</code>	This specifies the email address to which UPP is to send notifications.
<code>dist_node = fnkits.fnal.gov</code>	This specifies the product distribution node to contact.
<code>data_dir = /var/adm/upp</code>	This refers to the directory where you want UPP to maintain bookkeeping files.
<code>newprod_notify = T</code>	Setting <code>newprod_notify</code> to <code>T</code> (True) tells UPP to send notification of brand new products to the address in <code>mail_address</code> .

7.2.2 Identify the Product

Within a stanza, the following terms can be used in matching a new or updated product instance: `product`, `flavor`, `version`, `qualifiers`, `prod_dir` (product root directory), and `chain`. Set them to values that you want **UPP** to monitor on the distribution node.

*All instances that match a given set of values will be operated on (in contrast to the standard **UPS** and **UPD** matching algorithms; see Chapter 27: *Product Instance Matching in UPS/UPD Commands*). You can specify only the product name and thereby install all instances, or restrict the set of instances by specifying more information. Most of the time, you only need to specify `product` (and sometimes `flavor`). An example of this part of the file is:*

```
begin
  product = exmh
  flavor = SunOS+5.5
  ...
end
```

7.2.3 Trigger the Product Installation

After identifying the product instance, you need to tell **UPP** when to install it on your system. Your choices are when a new version of the product appears on the distribution node, or when the product on the distribution node gets chained to a value that matches your specification. This gets done in an *action* line, e.g.,

```
action = newversion
```

or

```
action = current
```

Any chain, including user-defined chains, can be specified.

7.2.4 Provide Instructions to UPP

At this point you're ready to tell **UPP** what to do when the conditions are met. Since this chapter discusses installing products, the instructions you can choose from are:

install	Install the subscribed product via upd install .
reget	Short for: delete, then reinstall
resolve	Run any ups declare commands as necessary to make chains match so that parent product and dependencies and run properly together.
notify	Place a notice of the new product instance in the mailed output.

7.3 Sample Subscription File for Installing a Product

This sample file instructs **UPP** to install all the SunOS+5.5 instances of the product **exmh** (and dependencies as necessary), and to resolve the dependencies. **UPP** is also instructed to send notification when the install is triggered. The file contents are on the left, and explanations on the right:

```
file = upp                This identifies the file as a UPP subscription file.

mail_address              = Send mail notifications to joe@fnal.gov.
joe@fnal.gov

dist_node                 = Use fnkits.fnal.gov as the UPD
fnkits.fnal.gov           server node to contact

data_dir                 = Use /var/adm/upp as the bookkeeping
/var/adm/upp              directory

newprod_notify = T       Yes (True), notify me of new products appearing
                           on the UPD server node (in this case, on the fnkits
                           node).

begin                     Begin stanza for a product.
```

product = exmh	Identify subscribed product as exmh (the exmh versions remain unspecified in this example, therefore act on all versions for the flavor specified below).
flavor = SunOS+5.5	Identify flavor of product (this is optional)
action = current	List in the following lines one or more functions to perform when an instance of exmh of flavor SunOS+5.5 is chained to current on <i>fnkits</i> .
notify	Send a notification message to <i>joe@fnal.gov</i>
install	Install the newly current instance (and its dependencies as necessary) on the local node
resolve	Determine which <code>ups declare</code> commands need to be run on the local node so that all the chains match up properly for the dependencies to work, then run the commands.
end	End. If you want to add instructions for another product in this same file, start a new stanza with "begin".

7.4 The UPP Command

The **upp** command line is very simple:

```
% upp [-v[v...]] <subscrip_file_1> [<subscrip_file_2>...]
```

The **-v** option requests verbose output; more **v**'s (up to four) provide progressively more verbosity. The **upp** command has no direct output (unless verbosity is turned on), rather it mails a report of any actions taken to the email address specified in the subscription file.

There are no other command options for **upp**; its behavior is controlled entirely by the subscription file(s).

7.5 Automating UPP via cron

You can add a **cron** job that first sets up **UPD** then runs **UPP** with a subscription file (shown here as `upp.subscription`). Here is a sample **sh** script to which we give the filename `upp.launch`:

```
#!/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
```

