Chapter 9: Installation-Specific UPS Database Files

This chapter describes the **UPS** database files that contain the product management information specific to a local **UPS** installation. In particular, it describes version, chain and **UPS** database configuration files. It concludes with a note for product installers regarding table files.

Product installers and **UPS** database administrators should be able to interpret these files, but in general reading them is necessary only in problem cases. Most of the time the **`ups list [-K] [-l]`** command provides sufficient information.

Product developers may need to read these files to determine that their products are installed and declared properly to the `KITS` or other **UPS** distribution database. In the case of `KITS`, **`upd list [-K] [-l]`** is usually sufficient.

9.1 Version Files

Version files get created via the **`ups declare`** command. One *version file* must exist 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., `v19_34.version`. It must reside in the product-specific directory under the **UPS** database directory, e.g., `$PRODUCTS/emacs/v19_34.version`. For a single version of a product, often a separate product instance is installed for each flavor; and sometimes more than one per flavor if qualifiers are used. Therefore, multiple product instances are often represented in a single version file.

A version file contains system-specific information for each instance, including (but not limited to):

- when the instance was declared
- who declared the instance
- the product root directory of the instance
- the location of the `ups` directory
- the location of the table file that is to be used for the instance (as described below)

A new version file is created automatically by **UPS** when the first instance of a new version of a product is declared to the **UPS** database. When a subsequent instance of the same version is declared, **UPS** automatically modifies this file to include information for it.

9.1.1 Examples

Sample version file for exmh v1_6_6: v1_6_6.version

```
FILE = version
PRODUCT = exmh
VERSION = v1_6_6

#*************************************************
```

```
#
FLAVOR = SunOS+5
QUALIFIERS = ""
  DECLARED = 1998-03-30 21.06.59 GMT
  DECLARER = stolz
  MODIFIED = 1998-03-30 21.06.59 GMT
  MODIFIER = stolz
  PROD_DIR = /afs/fnal.gov/products/UNIX/exmh/v1_6_6
  UPS_DIR = ups
  TABLE_FILE = exmh.table
```

This file was created when this instance of **exmh** was declared to the **UPS** database via a command like the following:

```
% ups declare -r /afs/fnal.gov/products/UNIX/exmh/v1_6_6 -m \
  exmh.table exmh v1_6_6
```

This assumes the machine is of flavor SunOS+5, the `ups` directory resides in its default location (directly under product root directory), the table file resides in a default location (see section 9.1.2) and we are using *$PRODUCTS* to determine the database.

Sample version file for foo v2_0: v2_0.version

Version files can contain information for multiple instances of a single version of a product. Here is an example for a fictional product **foo** v2_0. This file would have been created and modified by the series of commands:

```
% ups declare foo v2_0 -m v2_0.table -f IRIX -q superoptimize -r \
  /usr/prod/IRIX/foo/v2_0s
```

```
% ups declare foo v2_0 -m v2_0.table -f OSF1 -r \
  /usr/prod/OSF1/foo/v2_0
```

```
FILE = version
PRODUCT = foo
VERSION = v2_0

#*************************************************
#
FLAVOR = IRIX
QUALIFIERS = "superoptimize"
  DECLARER = aheavey
  DECLARED = 1998-04-15 16.37.58 GMT
  MODIFIER = aheavey
  MODIFIED = 1998-04-15 16.37.58 GMT
  PROD_DIR = /usr/prod/IRIX/foo/v2_0s
  UPS_DIR = ups
  TABLE_FILE = v2_0.table

 #---------------------------------------
 #
 FLAVOR = OSF1
 QUALIFIERS = ""
  DECLARER = aheavey
  DECLARED = 1998-04-15 16.39.58 GMT
  MODIFIER = aheavey
  MODIFIED = 1998-04-15 16.39.58 GMT
   PROD_DIR = /usr/prod/OSF1/foo/v2_0
   UPS_DIR  = ups
   TABLE_FILE = v2_0.table
```

9.1.2 Specification of Table File Location

UPS commands usually need to access the table file associated with the specified product instance, assuming one exists (most but not all products have one). **UPS** must therefore be able to determine the name and location of the table file. Any table file name and location information specified on a command line (via the `-m` and `-M` options; see section 6.3) overrides corresponding information in the version file. If TABLE_FILE is not defined in the version file (we recommend defining it there), and if it exists and should be used, it must be specified on the command line. If it is not specified either place, **UPS** assumes there is no table file.

In a version file, the table file name and location can be specified in several ways, depending on how the keywords have been defined. **UPS** uses the following algorithm to determine the table file location (note that TABLE_DIR and UPS_DIR can each be specified as either an absolute or a relative path; when specified as a *relative* path, **UPS** takes it as relative to PRODUCT_DIR_PREFIX/PRODUCT_DIR[1].):

If TABLE_FILE is specified as an absolute path, then:

- The location is TABLE_FILE.

If TABLE_FILE is specified as a relative path, or simply as the filename, then:

- If TABLE_DIR is specified, the location is TABLE_DIR/TABLE_FILE.
- If TABLE_DIR is not specified, and UPS_DIR is specified, then two locations are searched: first the product subdirectory in the database (e.g., `$PRODUCTS/my_product`), and second UPS_DIR/TABLE_FILE.
- If neither TABLE_DIR nor UPS_DIR is specified at all, **UPS** will search for TABLE_FILE under the product subdirectory in the database only.

9.2 Chain Files

Chain files get created via the **`ups declare`** ***chain_flag*** command. Chain files are named according to the chain name, and end with `.chain`, e.g., `current.chain`. A chain file's contents is simply a formatted list of the product instances that were declared with that chain. Each product instance is specified via a set of keywords. When a chain is used in a **UPS** command, **UPS** looks in the corresponding chain file to match the instance and thus locate the appropriate version file. As discussed in section 9.1, the version file entry locates the product root directory and table file to retrieve the instance.

In **UPS** commands, the command line option associated with a particular chain can be used to specify the product instance to retrieve. Using chains is optional, but recommended. Both chained and unchained instances of a product may be declared to **UPS**; the user can still retrieve any instance, chained or not, by specifying its version number.

Some examples of chain files follow.

Sample chain file for exmh v1_6_6: `current.chain`

This file points to the instance used in the first sample version file of section 9.1.1.

```
FILE = chain
PRODUCT = exmh
CHAIN = current
```

1. Be aware that PROD_DIR_PREFIX may not be defined; if not, PROD_DIR should be an absolute path.

```
#************************************************
#
FLAVOR = SunOS+5
QUALIFIERS = ""
  VERSION = v1_6_6
  DECLARED = 1998-03-30 21.06.59 GMT
  DECLARER = stolz
  MODIFIED = 1998-03-30 21.06.59 GMT
  MODIFIER = stolz
```

To setup the current instance of **exmh** on a SunOS+5 machine, which this file tells us is **exmh** v1_1_6, you would need to issue the command:

`% setup exmh`

The current instance is the default, so no other identifying information is needed in this case.

Sample chain file for foo: `current.chain`

This example illustrates the use of qualifiers. It points to both of the instances in the version file for **foo** in section 9.1.1. That version file will also get modified when these chains are declared. The DECLARER, DECLARED, MODIFIER and MODIFIED fields will include information for the chain declarations.

```
FILE = CHAIN
PRODUCT = foo
CHAIN = CURRENT
#
#----------------------------------------
#
FLAVOR = IRIX
QUALIFIERS = "superoptimize"
  VERSION = v2_0
  DECLARER = aheavey
  DECLARED = 1998-05-01 22.04.42 GMT
  MODIFIED = 1998-05-19 21.06.59 GMT
  MODIFIER = aheavey

FLAVOR = OSF1
QUALIFIERS = ""
  VERSION = V2_0
  DECLARER = aheavey
  DECLARED = 1998-05-05 17.04.42 GMT
  MODIFIED = 1998-05-24 21.06.59 GMT
  MODIFIER = aheavey
```

To setup the current instance of **foo** from an OSF1 machine, use the command:

`% setup foo`

The current instance is the default when no chain or version is specified. This sets up **foo** v2_0.

To setup the current instance on an IRIX machine, however, you would have to issue the command with the **`-q`** option:

`% setup -q superoptimize foo`

Note that if from IRIX you issue the command:

`% setup foo`

the command will fail. This is because the current chain on an IRIX machine has the additional qualifier *superoptimize* set, and this qualifier is necessary for instance matching.

This annoyance can be avoided by omitting qualifiers when declaring products. In lieu of declaring the qualifiers explicitly, installers can add comment lines to any of the product files to identify instances compiled with special options, as mentioned in section 2.1.3.

For this example in the IRIX case, the sequence of events upon issuing the **`setup -q ...`** command would be as follows:

1) match the FLAVOR (IRIX) and the QUALIFIERS (superoptimize) in this chain file

2) find the version (v2_0) and open the corresponding version file (`v2_0.version`)

3) locate the table file (`$FOO_DIR/ups/v2_0.table`) and open it

4) find the ACTION=SETUP line in the table file and execute the listed functions (if no ACTION=SETUP line were present, **UPS** would execute the default **`setup`** functions; see Appendix D)

9.3 UPS Database Configuration File: $PROD-UCTS/.upsfiles/dbconfig

☞ The **UPS** database configuration file contains system-specific information that customizes the **UPS** installation on a node or cluster. **UPS** does not require that a database have this file. The table in section 8.3 shows which keywords can be included in it. Here is a sample file:

```
FILE = DBCONFIG

AUTHORIZED_NODES = *

###SETUPS_DIR = /usr/local/etc
SETUPS_DIR = /tmp
UPD_USERCODE_DIR = ${UPS_THIS_DB}/.updfiles

MAN_TARGET_DIR = /usr/products/man
CATMAN_TARGET_DIR = /usr/products/catman
INFO_TARGET_DIR = /usr/products/Info

# PROD_DIR_PREFIX =
# STATISTICS = tcl,tclX,tclx,tk
# HTML_TARGET_DIR =
# NEWS_TARGET_DIR =
```

9.4 A Technical Note for Product Installers

Normally product installers never need to touch the table file (described in sections 11.1 through 11.7). Here is an exception.

If you have a product that needs to find *$MYPROD_DIR*, but **myprod** has been installed on your system with a different name, e.g., **fermi_myprod**, then you may need to edit the table file. If the provided table file for **myprod** was written by a developer who has no knowledge of the name change on your system, the table file probably contains:

```
ACTION=SETUP
   prodDir()
```

where `prodDir()` instructs the **`setup`** command to set the variable *<PRODUCT>_DIR* (see section 11.3*)*. On your system then, the variable *$FERMI_MYPROD_DIR* will get set, but *$MYPROD_DIR* won't. To ensure that you also get the variable *$MYPROD_DIR*, edit the table file and add the function:

```
envSet(MYPROD_DIR, ${UPS_PROD_DIR})
```

also under ACTION=SETUP.