

# Chapter 27: Product Instance Matching in UPS/UPD Commands

When a **UPS** or **UPD** command is issued, the system must determine which product instance(s) to act upon. This determination is called *instance matching*. This chapter describes the algorithms used for instance matching.

Instance matching for **UPS/UPD** is done within **UPS**. **UPD** commands can find the local **UPD** configuration, which is used to determine the local database for the command; beyond that, it relies on **UPS** to match the instance. A match is determined according to the database, and the product's name, flavor, version and qualifiers. You may specify all these parameters explicitly, in which case no special logic is required; the command will succeed or fail based on the given information. You are not required to specify all the information, however. At a minimum, you must provide the product name explicitly<sup>1</sup>. All the remaining information can be defaulted, thus requiring **UPS/UPD** to rely on preset algorithms to match the instance.

## 27.1 Database Selection Algorithm

---

### 27.1.1 UPS

To match a product instance, a **UPS** command must first determine the database in which to search for the product.

#### Database Not Specified on Command Line

If **-z <databaseList>** is not specified on the command line, **UPS** looks in `$PRODUCTS`. If `$PRODUCTS` points to a single database, **UPS** searches only that one. If `$PRODUCTS` points to multiple databases, then the following applies:

If the command is **ups declare** and it is being used to declare an instance to a database (not to declare a chain to a pre-existing instance) then **UPS**:

---

1. The commands **ups(d) list**, **ups flavor**, **ups help** and **ups(d) verify** can be used with no product name, version or options.

- looks at the `updconfig` file for the first database listed in `$PRODUCTS`,
- finds a matching stanza (the `updconfig` file uses a set of identifiers described in section 32.2 *Product Instance Identification and Matching*),
- and declares the instance to the database specified in the matched `updconfig` file stanza.

If a declaration for the instance already exists in the specified database, the command fails.

For all other commands (including **ups declare** when it is being used to declare a chain to a pre-existing instance), **UPS** searches the databases in the order listed in `$PRODUCTS`, and operates on the first matching instance it finds.

## Database Specified on Command Line

A database specified on the command line can be any database; it can be `$PRODUCTS` (which is equivalent to leaving off the database specification altogether), it can be one or more databases that are listed in `$PRODUCTS`, it can be one or more databases *not* listed in `$PRODUCTS`, or it can be a combination, e.g., `-z /path/to/somedb:$PRODUCTS`.

In fact, the construction `-z /path/to/somedb:$PRODUCTS` is frequently used (where `/path/to/somedb` may or may not be listed in `$PRODUCTS`). It allows you to give preference to a particular database by placing it first, while still letting **UPS** search the `$PRODUCTS` databases in the usual order for dependencies.

If `-z <databaseList>` is specified on the command line, and:

- `<databaseList>` specifies a single database, **UPS** uses that one.
- `<databaseList>` points to multiple databases, **UPS** searches the specified databases in the order given. It uses the same logic as described in section 27.1 *Database Selection Algorithm*. (Substitute `<databaseList>` for `$PRODUCTS` in the description.)



Note for product dependencies: If the database is specified on the command line via `-z <databaseList>`, then it gets propagated to all dependencies called using the functions `setupRequired()` or `setupOptional()` (these functions are described in Chapter 35: *Functions used in Actions*). This applies to both top level and lower level dependencies.

## 27.1.2 UPD

If your **UPS** installation includes only one database, that is the one **UPD** will use (assuming that the **UPD** configuration used by that database points to it, which is generally the case). If there are multiple local databases, the **UPD** command has to determine the database to use. In a nutshell, it:

1) picks a starting database

To pick the starting database, **UPD** first looks to see if the **-z <databaseList>** option is specified on the command line. If so, **UPD** picks the first database listed there. If not, **UPD** picks the first database listed in \$PRODUCTS.

2) finds the **UPD** configuration file to which the database points<sup>1</sup>

3) looks in this **UPD** configuration to see where to install, declare, update, etc., the product (using the UPS\_THIS\_DB location)



If your local **UPS/UPD** installation is particularly complicated, it might be useful to verify that your \$PRODUCTS variable includes all the databases used by the **UPD** configuration(s), and only those. If not, it is possible to declare products into a database not listed in \$PRODUCTS, which generally is undesirable.

## 27.2 Instance Matching within Selected Database

---

**UPS** uses a built-in algorithm based on flavor and qualifier matching in the product's version, chain, and table files to match the instance. Once a match is complete, resulting in a complete product identification, the instance is located and the command executes. The algorithm is described below. **UPD** does not do instance matching, it communicates with **UPS** for this purpose.

### 27.2.1 Where Does Instance Matching Take Place?

Instance matching takes place in the product's version, chain, and table files. Chain files point to version files which point to table files (this is all described in Chapters Chapter 29: *Version Files*, Chapter 30: *Chain Files* and Chapter 36: *Table Files*), and the instance must be matched at each stage before moving on.

---

1. The location of the `updconfig` file is set via the keyword `UPD_USERCODE_DIR` in the database's `dbconfig` file.

The point at which the process starts depends on what is specified on the command line:

- If a chain is specified on the command line (and no version), then **UPS** first matches the instance in the corresponding chain file. If that chain file doesn't exist, the command fails.
- If no chain or version is specified, then the current chain is assumed by default, and **UPS/UPD** first matches the instance in the `current.chain` file. If this file doesn't exist, the command fails.
- If a version is specified (and no chain), then **UPS/UPD** first matches the instance in the version file. If the specified version file doesn't exist, the command fails.
- Regardless of whether a chain or version was specified, if a table file exists and is specified in the version file, **UPS/UPD** does a final instance matching in the table file. If no match is found, the command fails. If a table file is specified on the command line, **UPS/UPD** bypasses the version and/or chain files, and matching takes place only within the table file.

## 27.2.2 Flavor Selection

You can specify the flavor on the command line using `-f` or one of `-0`, `-1`, `-2`, `-3`, or `-H` (alone or together with one of `-0`, `-1`, `-2`, `-3`). These options are described in Chapter 25: *Generic Command Option Descriptions*. If you do not specify a flavor, **UPS/UPD** determines the flavor based on the set of rules given in section 27.2.4 *Flavor and Qualifier Matching Algorithm*. In most cases **UPS/UPD** can determine the flavor successfully.

Multiple flavors can be processed for many of the **UPD** commands, and for the **UPS** commands `ups list` and `ups verify`.

### 27.2.3 Qualifiers: Use in Instance Matching

If the option **-q** is included in the command with qualifiers as arguments, each qualifier can be specified in one of two ways:

- required qualifier (no special characters, e.g., **-q optimize**); command will fail if this qualifier cannot be matched
- optional qualifier (preceded by **?**, e.g., **-q "?debug"**); **UPS/UPD** will try to match this qualifier but command will *not* fail if it cannot be matched

Multiple qualifiers can be specified; mixing required and optional is allowed.

### 27.2.4 Flavor and Qualifier Matching Algorithm

If a flavor and/or a qualifier list is specified on the command line, **UPS** attempts to match the information exactly. The command fails if an exact match cannot be found (except for optional qualifiers). If no qualifiers are specified, no instance with qualifiers will be selected as a match. If the flavor is not specified or if **-H** is used (alone) to generate a “host” flavor list, **UPS** selects the first matching instance it encounters according to this algorithm:

- 1) That instance with a flavor exactly matching the machine’s or host `<OS_type>+<OS_version>` and matching qualifiers, if any.
- 2) That instance with a flavor exactly matching the machine’s or host `<OS_type>` and the OS version specification which is the longest first substring of `<OS_version>`, and matching qualifiers, if any.
- 3) That instance with a flavor exactly matching the machine’s or host `<OS_type>` and having no release specification, and matching qualifiers, if any.
- 4) That instance with a flavor of `NULL`, and matching qualifiers, if any.

The selection fails if none of the above can be met within the determined database.



Be aware that the flavor `ANY` is allowable in a table file, and as expected, it will serve as a match for any flavor, even if the actual flavor appears later in the file. A flavor list generated by **-H** includes the value `ANY`.

