NetView Distribution Manager/2
Solving the Mystery of Global Names

June 17, 1994

NetView DM/2 Development Team
contact:Benny Giangregorio
GIANGRE @ ROMEPPC

Rome Networking Software Laboratory
via P. Di Dono, 44
00143 Rome
• 39(Italy)-6(Rome)-596(IBM)-62587

Unclassified

Warning

Do not remove any pages from this printed copy.

Read this first!

Most of this document is obtained by putting together the information that is already available in the NetView DM/2 2.0 publications; it aims at collecting all the information in one single reference document on that subject from a functional point of view.

Neither this document is intended to enhance the current level of NetView DM/2 publications, nor there is any warranty to update this document when a new refresh of the NetView DM/2 books is made available.

Document History

Date (revision bar character) Reason

June 17, 1994 First issue

Contents

1. Solving the Mystery of Global Names

Read this first!

Most of this document is obtained by putting together the information that is already available in the NetView DM/2 2.0 publications; it aims at collecting all the information about the global naming convention in one single reference document on that subject from a functional point of view.

Neither this document is intended to enhance the current level of NetView DM/2 publications, nor there is any warranty to update this document when a new refresh of the NetView DM/2 books is made available.

Introduction

This document contains information related to global names, change files and aims at giving some suggestions about the naming convention to use when creating the change files to do Change Management in your distributed workstation environment.

An object is used by NetView DM/2 to distribute data and install software. Each object has information saved in the catalog. An object is generally identified by:

- A global name that uniquely identifies the object in the network.
- The local file name of a local file that contains the object's data; this name follows the naming convention of the file system of the operating system in the workstation where it resides.

You need to prepare an object when:

- You have user data that must be sent in the network.;
- You want to install new software
- You want to apply a change to existing software
- You want to install or update a relational table

What is a Change File

A change file is used where the ultimate purpose is to install software or data on a CC single node or in a LAN environment on a CC client workstation. It specifies how an object is to be installed on the target workstation, and can be used in a different ways, depending on the installation methodology used.

To build a change file, you must provide three different types of information that control the installation process:

- Catalog information. Identifies the type and global name of the object and its processing size.
- File specification information. Provides the local workstation names of the files that are to be built into the change file. The files of an object do not necessarily need to be specified here if they can be identified by the installation process in some other way (as described in “Installing Change Files” on page 1-2).
- Installation information. Identifies how the object is to be installed (also described in more detail in “Installing Change Files” on page 1-2).

You can build a change file using the CDM dialogs or the CDM BUILD command. With the BUILD command you create a *change file profile*, using any editor that can create an ASCII format output file.

Use NetView DM/2 to prepare change files at one workstation, and distribute and install them in a wide area network using APPC links. Then use NetView DM/MVS to plan and coordinate this activity.

Installing Change Files

You can install change files in two ways:

Replication installation
With this method you replicate on a target workstation a manual installation of software. At the preparation site, the attended installation of software places some files in certain directories and updates other files. The files that are installed are then built into a change file that is distributed to the application sites. To install at the application site, CDM first unpacks the change file that contains the files. CDM replicates the structure and contents of the directories at the preparation site. CDM then runs a postinstallation program, if required, to update specific files; for example, the CONFIG.SYS file.

Response file-driven installation
Response files enable CDM to use the installation program supplied with software packages to carry out the task of installing the software on unattended workstations. The package preparer defines a set of responses to the prompts from the installation program and stores them in the response file.

You copy the response file and the files in the software package into a shared area on the CC Server workstation. At the target workstation, CDM then starts the installation program, which uses a response file to install the software on the designated target directory.

The installation program for the software can also customize the workstation for the object, for example, by updating the CONFIG.SYS file and the desktop.

The end result of both methods is to have the object installed in the directory of your choice, and the appropriate system configuration files updated.

Global Names to identify a Change File

Global names are used by CDM to uniquely identify objects.

A global name is made up of a string of *tokens* separated by periods. When you record an object in the catalog, you must specify the tokens in its global name.

Global names can have between two and 10 tokens, each containing up to 16 alphanumeric or special characters. A period is used to separate the tokens and the total length of the global name must not exceed 64 characters, including periods. The character set used is one that is supported on a wide variety of systems and display devices.

You use free-format tokens to label the catalog entry in a way that makes an object easy to identify; but if you have a systematic naming convention, that helps you keep track of the software installed on your workstations.

If the object is a change file, you use a fixed-format token (REF, UPD, or FIX), called the *change type* token, for one of the tokens. The change type token can vary its position within the global name with some restrictions.

You can use numeric tokens to sequence catalog entries that have otherwise identical global names. For example, you can direct CDM to find either the oldest or the newest version of an object.

When an object is distributed by NetView DM/2, the catalog is updated with the last change date of the local files in the object. This prevents a changed object being redistributed under its old global name.

For objects that are not to be used for making changes to production software or data, the tokens can follow any convenient local conventions, as long as you do not use the change type token values.

Advantages of a Systematic Naming Convention for Change Files

As you have noticed, NetView DM/2 uses the objects as a generic vehicle to to perform common software maintenance activity on a workstation such as:

- Install new software (either application or system software)
- Refresh previously installed software with a new release
- Upgrade previously installed software to a new level (or release)
- Apply a CSD
- Apply a fix

Generally, when you are performing one of these tasks, you are changing the software on a workstation; that is the reason why those objects are called *change files*.

NetView DM/2 can change an application, an operating system, or a smaller component of these.

Without NetView DM/2, when you want to change one of the applications or the system software on your workstation, you:

- Check whether the change can be applied according to the current level of software.
- Apply the change to the same directory where you have previously installed the software.

If you have implemented a systematic naming convention for the objects, NetView DM/2 can automatically perform the following tasks for you:

- Identify the software to be changed, according to the structure of the global name of the object.
- Check whether the level of the change is compatible with the current level of the existing software.
- Install the change in the same directory where the software was previously installed.
- Keep track of the change, by updating the information about the level of the software currently installed on your workstation.

In order to get the best results from these NetView DM/2 functions, the naming convention must match the one used for the *component name* and the *installation target*, as described in “Component Names are used to identify affected Software Component” on page 1-7 and “Component Names are used to identify common Installation Path” on page 1-10.

Note: The underlying assumption is that you always use NetView DM/2 to install and update software.

Global Names character set

In global name tokens, you can specify only the following characters:

- Uppercase alphabetic characters (A – Z)
- Numerics (0– 9)
- These special characters:
  - **$** dollar sign
  - **#** number sign
  - **@** at sign

_ underscore

Using Special Tokens

The following are the special tokens that can be used:

&netid
: This token specifies the network ID of the local system. You enter it in the global name in the format shown here, but it is translated into the appropriate value when you use the catalog or build actions to add an object to the catalog.

&luname
: This token specifies the logical unit name of the local system. It is entered in the global name in the format shown here, but is translated into the appropriate value when the object is added to the catalog by the catalog or build actions.

&datetime
: This token specifies a date and time value. It is entered in the global name as it is shown here, and is incorporated into the catalog entry by the catalog or build actions in the same format. It can be specified only as the last token in the global name.

The &datetime token is *never* retained as a token in the global name of an object being transmitted in the network. Instead, before an object is sent by CDM, the &datetime token is replaced by the date and time from the OS/2 directory, which is the date and time when the object was created or last changed.

If a retrieve request is received from a focal point, CDM gives a positive match for any numeric token in the same position as the &datetime token.
For example, an object is cataloged with the global name:

```
AAA.BBB.CCC.&datetime
```

To match with this global name, the retrieve request would have to specify the following global name as a parameter to the request:

```
AAA.BBB.CCC.930401000000
```

The example shows a particular value for the &datetime token, but any value in this token is acceptable.

Note: Each time a global name includes the &datetime token, it must be enclosed in double quotation marks.

You can use numeric tokens to sequence catalog entries that have otherwise identical global names. For example, you can direct a CDM search routine to find the object with the lowest value (L) or highest value (H) of the numeric token, so that you can find the oldest or most recent version of the object.

For objects that are not to be used for making changes to production software or data, the tokens can follow any convenient local conventions.

The global names of change files are restricted by other conventions, which are described for each type of change file in the section "Global Names for Change Control."

Global Names for Change Control

The global names of change files and relational data objects must conform to a standard format.

When you prepare a change file to change the software of a workstation, you must associate a global name with it (by cataloging the object) in order to describe the type of change that the object represents. For this reason a particular token, the *change type token*, is inserted in the global name (see “Change Name” on page 1-5).

Change files and relational data objects are identified by global names that consist of a *component name* and a *change name*, consisting of a *change type* token and its associated tokens. The global names of all objects other than change files and relational data objects are free-format, except for the first token, which is recommended to be the corporation ID (see “Component Name”).

The structure of the component name and of the change name for each change type is described in the following sections.

Component Name

The component name is free-format, but it is recommended that you adopt a convention that has been developed for change files, whereby the first two tokens of the component name are:

corpid.compid

where:

corpid Identifies the corporation responsible for the component, usually by the registered company name or its abbreviation. For example: EUROTRAVEL.

The recommended way to specify a company name is described in the *SNA/Management Services Reference* manual, under “Enterprise Structured Names.” Part of the recommended name is derived from a structured network ID that can be registered by an authorized IBM representative using the SNAREGISTRY application on the HONE system. By obtaining a registered structured network ID, you can specify a company name that is unique. This is useful if objects that were created by different companies are exchanged in the network.

compid Identifies the overall system component or product, for example, a program number or a generic name. For example, if you specify IBM as the corpid, you could specify a seven-character IBM program number as the compid.

Example of corpid and compid:

EUROTRAVEL.HOLS

For relational data objects, the last three tokens of the component name are interpreted as the default name of the local database (one token) followed by the name of a table or view (two tokens). This latter name consists of a qualifier indicating the owner of the table (usually a user ID) and the table name itself. This means that the component name of a relational data change file must have at least five tokens. For further information, see your database manager documentation.

Change Name

The leading token of the change name identifies the type of change to be made. This is called the change type token, and it determines which tokens follow it.

The following is a list of the valid change types and the corresponding change type token you can use:

- Refresh (REF)

- Update (UPD)
- Fix (FIX)

These change type tokens cannot be used in the global name of an object that is not a change file, because CDM processes an object as a change file when these tokens are specified.

When you specify the change type token, NetView DM/2 checks for the software level and version validity when installing the change (as discussed later in this section).

The tokens that follow the change type give additional information about the change, and must follow rules dependent on the change type.

REF Object: A REF object installs a component for the first time or completely replaces a whole component that has previously been installed. In the case of relational data, a REF object replaces one table in a database. The change name has the following form:

REF.level.version.&datetime

where the meanings of the tokens are as follows:

level A numeric value that shows the maintenance level of the refresh

version (Optional) One or more tokens distinguishing objects at the same maintenance level

&datetime (Optional) An OS/2 date and time

Example:

EUROTRAVEL.HOLS.MESSAGES.REF.13.ENGLISH.020390151720

UPD Object: A UPD object is used to upgrade components to a new maintenance level. In the case of relational data, a UPD object appends data to an existing table in a database. In the case of all other change files it replaces the files indicated in the catalog. The change name has the following form:

UPD.oldlevel.newlevel.version.&datetime

where the meanings of the tokens are as follows:

oldlevel A numeric value that shows the minimum installed level of a component.

newlevel A numeric value that shows the upgraded level of the component. The newlevel value must be higher than the oldlevel value.

version (Optional) One or more tokens distinguishing objects at the same maintenance level.

&datetime (Optional) An OS/2 date and time.

Example:

EUROTRAVEL.HOLS.MODULES.UPD.12.14.BRANCH4.041490110345

FIX Object: A FIX object is used to fix a specific problem in a component. In the case of relational data, a FIX object appends data to an existing table in a database. In the case of all other change files it replaces the files indicated in the catalog.

Note: A FIX and a UPD object update the target data in the same way. However, you should use each type for its indicated purpose as a form of documentation. The change name has the following form:

FIX.level.probid.version.&datetime

where the meanings of the tokens are as follows:

level The maintenance level to which the fix applies

probid An identifier for the problem being fixed

version (Optional) One or more tokens distinguishing objects at the same maintenance level

&datetime (Optional) An OS/2 date and time

Example:

EUROTRAVEL.HOLS.MODULES.FIX.12.PROB036.BRANCH4.051990144001

Change Type and Object Type Correspondence

The following table shows you which change types are applicable for different types of objects, and summarizes the change name structures.

Table 1-1. Change Type Rules

Change name structure	Object types	Notes
REF.level.version.&datetime	Software Flat data Relational data	1,2
UPD.oldlevel.newlevel.version.&datetime	Software Flat data Relational data	1,2,3
FIX.level.probid.version.&datetime	Software	2

Notes:

1. The component name for a relational data object must contain at least five tokens.
2. The component name for a change file must contain at least two tokens.
3. The oldlevel value must be lower than the newlevel value.

Component Names are used to identify affected Software Component

NetView DM/2 checks the component name portion of a global name in order to identify which part of a software product is affected by the installation of a change file.

NetView DM/2 interprets the global name tokens according to their hierarchical position, therefore, a component name token provides more information on the software component that was specified by previous tokens.

The component name must be at least two tokens long; these two minimal tokens identify the software product.

The semantic value of the token positions must be taken into consideration when you decide the naming convention of the software products you want to install via NetView DM/2. For instance, suppose that the EUROTRAVEL company develops an application called FINANCE2. The features of this application can be divided into components, as illustrated by the following NetView DM/2 global names:

EUROTRAVEL.FINANCE2.ACCOUNT.REF.100
and
EUROTRAVEL.FINANCE2.ACCOUNT.DATA.UPD.100.102

where FINANCE2 is the software name and ACCOUNT is the feature name.

On the other hand, the following example does not match the meaning that NetView DM/2 gives to that change file:

EUROTRAVEL.FINANCE2.V50.INSTALL.REF.1

The INSTALL token in this example does not specify which part of FINANCE2 will be changed; rather it describes why the change file was created. Considering the V50 token, if you want to specify the version and release of the software, that information can be in the second token of the component name or in the LEVEL token, as shown in the following examples:

EUROTRAVEL.FINANCE2_V50.REF.0
or
EUROTRAVEL.FINANCE2.REF.500

See "Level Tokens in an Change File and Current Level of Software" for more information.

The first column of the following table shows examples of component names which do not follow NetView DM/2 semantic rules. The second column shows possible adjustments required for the name to be correctly interpreted by the product during the installation checks:

Table 1-2. Examples of Component Name Checking

Component Name	Corrected Name
IBM.OS2V20.SEINST.PRISTINE	IBM.OS2V20.
IBM.OS2V20.SEMAINT.	IBM.OS2V20.MAINT.
CORP.PROD1.XCOPY.DLLS	CORP.PROD1.DLLS

In general, when naming a change file, you should try to describe what the change file adds or updates. Do not use tokens to describe how the change file works or when it must be used.

For CID-enabled products, you need to generally prepare three change files in order to install a product. These change files install the following:

- Diskette images
- Response files
- Product with a configuration

Using the Communications Manager/2 product as an example, a suggested naming convention is the following:

- IBM.CM2_V1.DISKIMGE.REF.0
- IBM.CM2_V1.RSPFILES.<server_name>.REF.0
- IBM.CM2_V1.PRODUCT.REF.0

Although these three change files refer to the same software, you can maintain them separately. Therefore, if you add new response files, you will have a file called IBM.CM2_V1.RSPFILES.<server_name>.UPD.0.1 that installs IBM.CM2_V1.PRODUCT.REF.0 from IBM.CM2_V1.DISKIMGE.REF.0.

Level Tokens in an Change File and Current Level of Software

OS/2 and DOS system and application software is generally distributed on diskettes as:

- A new version or release
- An upgrade from one release to another
- A Corrective Service Diskette (from one service level to another)
- A fix to a problem (to be applied to software at a certain level)

If the software is installed as a NetView DM/2 object, the level is specified in:

- The first token after the REF change type token
- The second token after the UPD change type token
- The first token after the FIX change type token

The current level of a software product is the level specified in the global name of the last object installed. Its status must be active and its component name must match the one specified for the installation target name (see “Component Names are used to identify common Installation Path” on page 1-10).

For example, if you first install:

MICROSOFT.MSWINDOWS.REF.30

and then:

MICROSOFT.MSWINDOWS.UPD.30.31

the current level of software for MICROSOFT.MSWINDOWS software is *31* (see “Version and Maintenance-Level Constraints” on page 1-12 for more information).

Bearing this in mind, you can set a default global name by deciding what is to be considered as the level of software installed using NetView DM/2.

For example, the level of software can be the version or the release number of a product. The following can be the global name of a change file installing version 3.0 of MS/Windows:

MICROSOFT.WINDOWS.REF.30

In this case, the following defines a change file installing version 3.1 of MS/Windows by completely replacing the previous version:

MICROSOFT.WINDOWS.REF.31

In contrast, the following global name defines a change file installing only those new files that upgrade version 3.0 of MS/Windows to version 3.1:

MICROSOFT.WINDOWS.UPD.30.31

After installing this object, NetView DM/2 tracks in the catalog that REF.30 plus UPD.30.31 are currently active.

For IBM software you can also put the version or the release in the component name and use the service level as the software level.

The service level is the number of Corrective Service Diskette (CSD) currently applied (the same that you get by entering the SYSLEVEL command in an OS/2 session). The following example defines a change file installing the NetView DM/2 2.0 diskette files refreshed to CSD level 1:

I3IBM.4567123.NETVIEWDM2_V20.CODEIMAGES.REF.1

In contrast, the following example defines a change file installing the NetView DM/2 2.0 diskette files refreshed to CSD level 2:

I3IBM.4567123.NETVIEWDM2_V20.CODEIMAGES.UPD.1.2

The same considerations apply to system software generally installed using the response file method.

For example, the following global name defines the change file that installs OS/2 2.0 by running the response file-driven installation program SEINST.EXE:

I3IBM.1231230.OS2.20.BASE.REF.0

Then, the following can be the global name of the change file that installs the CSD XR6055 on top of the base OS/2 2.0, by running the response file-driven installation program FSERVICE.EXE:

I3IBM.1231230.OS2.20.BASE.UPD.0.6055

As in the previous example, NetView DM/2 keeps track of changes. In this case, it tracks that REF 0 plus UPD from 0 to 6055 are currently installed.

Versions and Time Stamps for Change Files

When assigning global names to change files you can add a final optional set of tokens giving the version and date and time of the software or data. These distinguish different files at the same maintenance level.

The time-stamp token must be the last token. You should enter it using the &datetime token. When converted to an OS/2 date and time, the time stamp has two digits each for month, day, year, hour, minutes, and seconds, as follows:

MMDDYYHHMMSS

These parameters are important for national language support and customizing purposes. For example, a version token can indicate the language in which a help window is written. Another example is customizing a software application for several users, in which case a version token would indicate the user by name.

Here is an example of a Eurotravel REF object, where the change is of a list of special offers. In this case the holidays are in Italy, so the version is ITALIAN. A time stamp is also present.

EUROTRAVEL.HOLS.SPOFFERS.REF.13.ITALIAN.010291095750

When a catalog entry ending with an &datetime token is used to locate a matching local file, the &datetime token is ignored. If the matching file is to be transferred to another node, the date and time of the last update on it, which is recorded in the OS/2 directory, is inserted as the last token of the global name, in place of the &datetime token, before the file is transmitted.

Component Names are used to identify common Installation Path

Strict use of a systematic naming convention for the change files can free you from having to specify the target directory every time you want to apply a change to the same software.

When you install the first change file for a software product, NetView DM/2 records the software location in the catalog.

To maintain the installed software, you should install all the related changes in the same target directory. During installation, NetView DM/2 creates a mapping between a subset of the tokens of the component name of the first change file installed (the installation target name) and the target directory where the software has been installed. This mapping is called the **installation target**.

In this way, all the subsequent changes for that software are installed in the same directory, regardless of the target directory specified in the change file.

Note: If you do not want to use this function, you can delete the mapping and install the object in the directory that you specify.

Predefined installation targets

Installation targets can be predefined explicitly at a target workstation by a local user, using either CDM dialogs or commands. In this case, any default path information carried in a change file is ignored by CDM during the installation process, and the predefined installation target is used instead.

During installation of the change file, CDM looks for the default path information, which you can optionally specify with the *TargetDir* parameter when you build the change file. CDM automatically defines an installation target for that component at the target workstation using the default information in the change file. From then on, any change file that has that component name in the first part of its global name is installed using that path information, unless the local user redefines it.

For example, suppose that an installation target (fully qualified directory path) is predefined for a component name:

OS2.COMMGR.APPC

When you then try to install an object with the global name:

OS2.COMMGR.APPC.DLL.FIX.2.10105XR

CDM installs it in the same directory as indicated by the installation target already defined in the Catalog, because the component name part of the global name of that change file matches the component name in the catalog.

Automatically defined installation target

The installation target name is based on the global name specification, and it uses the following naming conventions:

- If you do not specify a value in the **Component Name Length** field in the Installation window (or CompNameLen key in the change file profile), the name is composed of all tokens up to (but not including) the change type token.

  For example, if you install OS/2 2.1 for the first time and use the global name:

  I3IBM.1231230.OS2V21.BASE.REF.0

  the installation target name is created in the catalog:

  I3IBM.1231230.OS2V21.BASE

- If you do specify a value for **Component Name Length**, the installation target name is composed of the number of tokens specified.

  For example, if you install OS/2 2.1 for the first time with the following global name:

  I3IBM.1231230.OS2V21.BASE.REF.0

  and with **3** as **Component Name Length**, the installation target name is placed in the catalog:

  I3IBM.1231230.OS2

In addition, the following rules apply:

1. If the component name of a change file is the same as an installation target name already present in the catalog, NetView DM/2 installs the change file by default in the target directory pointed to by the installation target name.

   For example, if the following installation target name is present in the catalog:

   I3IBM.1231230.OS2V21.BASE

a change file with the global name:

I3IBM.1231230.OS2V21.BASE.FIX.0

is installed by default in the target directory pointed to by the installation target name, even if the target path is not specified.

2. If the component name of a change file contains an installation target name already present in the catalog, NetView DM/2 installs the change file by default in the same target directory pointed to by the installation target name. This is because the object is assumed to identify a change to a subset of that software.

   For example, if the following installation target name is present in the catalog:

   I3IBM.1231230.OS2V21.BASE

   a change file with the global name:

   I3IBM.1231230.OS2V21.BASE.DLL.FIX.0

   can be installed in the target directory pointed to by the installation target name, even if the target path is not specified.

3. If the component name of a change file is contained in an installation target name already present in the catalog, you must specify a new target directory, because the software to be installed is likely to be new or more general.

   For example, if the following installation target name is present in the catalog:

   I3IBM.1231230.OS2V21.BASE

   a change file with global name:

   I3IBM.1231230.OS2.REF.1

   cannot be installed in the target directory pointed to by the installation target name, unless you specify the target path in the change file profile.

See “Global Names and Secondary Effects during installation” on page 1-14 for further details about the rules to apply when defining objects.

Version and Maintenance-Level Constraints

There are some constraints when installing refresh, update, and fix objects, based on the following assumptions:

- If an object has a component name that includes the component name of another object, the former is assumed to be a change of a more specific software component than the latter. For example:

  IBM.OS2.V2.COMMGR.APPC.DLL.REF.0

  is a subcomponent of

  IBM.OS2.V2.COMMGR.REF.0

- The current installed level of a software component is the highest level of all the objects installed in production and having the same component name. For example, if you have installed:

  IBM.OS2.V2.COMMGR.REF.0

  and

  IBM.OS2.V2.COMMGR.UPD.0.5

  the current installed level for IBM.OS2.V2.COMMGR is 5.

Installing REF Objects: The version of a REF object to be installed must match the versions of all installed objects with matching, shorter component names, whenever they have version tokens in their global names.

Thus, if you have already installed:

IBM.OS2.V2.COMMGR.REF.1.ENGLISH

you cannot install:

IBM.OS2.V2.COMMGR.APPC.REF.2.ITALIAN

because the version is different.

Thus, a refresh of a software subcomponent to be installed must have the same version of the most comprehensive component installed.

Installing UPD Objects: The version of a UPD or FIX object to be installed must match the versions of all installed objects with matching, equal-length component names.

Thus, if you have already installed:

IBM.OS2.V2.COMMGR.REF.3
IBM.OS2.V2.COMMGR.UPD.3.4.ENGLISH

you cannot install:

IBM.OS2.V2.COMMGR.APPC.UPD.4.5.ITALIAN

because the version is higher.

Thus, a fix or upgrade of a component must have the same version of the software previously installed. upgrade from a level lower than or equal to the current installed level of the same software component.

Thus, if you have already installed:

IBM.OS2.V2.COMMGR.UPD.1.2
IBM.OS2.V2.COMMGR.DLL.FIX.2.PROB1245XR
IBM.OS2.V2.COMMGR.REF.3

you cannot install:

IBM.OS2.V2.COMMGR.UPD.4.5

because the old level of the new object (4) is higher than 3 (the highest among the levels of the first and third object).

The new level of the UPD object must be higher than the highest level of all objects with a matching, longer component name.

This is because an UPD object is always assumed to contain an upgrade to a level higher than the current installed level of any more specific components.

Thus, if you have already installed:

IBM.OS2.V2.COMMGR.APPC.UPD.3.4

you cannot install:

IBM.OS2.V2.COMMGR.UPD.2.3

because the new level of the later object is not higher than that of the existing object.

Installing FIX Objects: When you install a FIX object, the level of the FIX object must be the same as the highest level of all objects with a matching, equal-length component name, and this must also be the same as the highest level of all objects with a matching, longer component name.

This is because a FIX is always assumed to be made to the latest version of software installed, even if that installation was only to part of the software item.

Thus if you have already installed:

IBM.OS2.V2.COMMGR.APPC.REF.2.ENGLISH

and:

IBM.OS2.V2.COMMGR.APPC.DLL.REF.3.ENGLISH

you cannot install:

IBM.OS2.V2.COMMGR.APPC.FIX.2

because a more specific component at level 3 has been installed.

Global Names and Secondary Effects during installation

NetView DM/2 assumes that if you are installing a software change, previous changes are overridden on the hard disk. Therefore, it updates the tracking information on the catalog to reflect the new situation on the workstation.

This implies two important secondary effects of installing UPD or REF objects nonremovably:

- Any FIX object for the same software component previously installed nonremovably is:
  - Tracked as no longer installed, or
  - Changed into BKL status if the installation option specifies removability.

  This is because a refresh or upgrade of a software component overrides the installation of previous fixes.

  For example, if you have the following catalog entries:

  OS2.V2.COMMGR.REF.0
  OS2.V2.COMMGR.FIX.0.PROB34

  and you are installing

  OS2.V2.COMMGR.UPD.0.3

  the result in the catalog is the following:

  OS2.V2.COMMGR.REF.0
  OS2.V2.COMMGR.UPD.0.3

- Any UPD objects for the same component having an old level and a new level that are included between the old level and the new level of the object being installed are:
  - Removed from the catalog, or
  - Changed into BKL status if the installation option specifies removability.

  This is because the installation of a new level of a software component overrides the installation of upgrades to a minor level.

  For example, if you have the following catalog entries:

  OS2.V2.COMMGR.REF.1
  OS2.V2.COMMGR.UPD.1.3

  and you are installing

OS2.V2.COMMGR.UPD.0.4

the result in the catalog is the following:

OS2.V2.COMMGR.REF.1
OS2.V2.COMMGR.UPD.0.4

- Any REF object for the same component having a lower level than the one of the current REF object installed is:
  - Removed from the catalog, or
  - Changed into BKL status if the installation option specifies removability.

  For example, if you have the following catalog entries:

  OS2.V2.COMMGR.REF.0
  OS2.V2.COMMGR.FIX.0.PROB125
  OS2.V2.COMMGR.UPD.0.1

  and you are installing

  OS2.V2.COMMGR.REF.1

  the result in the catalog is the following:

  OS2.V2.COMMGR.REF.1

Secondary effects are listed in the report of the install action sent to the focal point.

A sample scenario about the usage of Global Names

The following section illustrates a sample scenario where a fictitious company has to maintain a set of applications. As NetView DM/2 is used to acomplish the installation and maintenance, the scenario shows the global names the company uses to take advantages of the Change Mangement checking and tracking inside NetView DM/2 product.

Note: All the non-IBM names in this scenario are fictitious and they are not intended to refer to existing products.

Scenario description

The EUROTRAVEL company has to install a new pristine workstation with OS/2, LAPS, NetView DM/2 and Communications Manager/2 1.1 system software.

Then he needs to install a vendor supplied and an internally developed. The former - TRAVEL OFFICE/2 - consists of 3 separately installable features:

1. Electronic Mail feature
2. Word Processor feature
3. Business Account feature

Then latter - TOUR-GUIDE/2 - has only one feature but it requires two tables in order to run. The two tables are not relational tables in this example. They contains

- Special Offers
- Prizes

and are periodically updated when a change in the Special Offers table or in the Prizes table occurs.

Then when the Word Processor application needs a fix, the company policy states that if the fix is provided by the vendor company, the fix is installed until a new release of the TRAVEL OFFICE/2 application or an upgrade of one its feature is made available. Anyway, an upgraded of a feature is installed only if it is compatible with the previous release if the other features.

Subsequently, EUROTRAVEL decides to install the first OS/2 2.1 CSD and when a new CSD of NetView DM/2 is made available, they decide to install that.

Adopted Global Naming convention

1. To install a pristine workstation with the basic system software, EUROTRAVEL company uses the following global names:

Adopted name	Considerations
IBM.OS2V21.REF.0	The “level” token is the maintenance level of that release.
IBM.LAPS.REF.100	The “level” token is the version, release and maintenance level of that release.
IBM.NVDM2_CLIENT_V20.REF.5	The “level” token is the maintenance level of that release The product is installed from a manifacturing refresh of the package which is already updated with CSD 5.
IBM.COMMGR2.REF.110	The “level” token is the version,release and maintenance level of that release.

2. To install the 3 feature of the TRAVEL OFFICE/2 application, the EUROTRAVEL company prepares 3 separate change files with the following global names: (note that the application is the 3rd release according to the EUROTRAVEL internally adopted release number)

Adopted name	Considerations
EUROTRAVEL.TRAV_OFFICE2.MAIL.REF.3	The “level” token is the EUROTRAVEL internally adopted release of the application; The 3rd token distinguish one feature from the others.
EUROTRAVEL.TRAV_OFFICE2.WORD_PROC.REF.3	The “level” token is the EUROTRAVEL internally adopted level for the application. It shares the first 2 two tokens with the other 2 change files in this table.
EUROTRAVEL.TRAV_OFFICE2.BUSINESS.REF.3	The “level” token is the EUROTRAVEL internal adopted level for the application. The 3 features have the same level.

3. To install the TOUR GUIDE/2 application and the “Special Offers” and “Prizes” files for the first time, EUROTRAVEL company prepare 3 change files with the following global names:

Adopted name	Considerations
EUROTRAVEL.TOURGUIDE2.APPLICATION.REF.1	The level token is internally adopted by EUROTRAVEL company to mean the release of the application.
EUROTRAVEL.TOURGUIDE2.PRIZES.REF.1	The “level” token is meant to indicate the update level of that table and it is not related to the level of the application. Only during the firsdt installation it is equal to the level of the application; then it will be increased every time there is a new PRIZES list. the “level” token of the first object.
EUROTRAVEL.TOURGUIDE2.SPECIALS.REF.1	The “level” token is meant to indicate the update level of that table and it is not related either to the level of the application or the Prizes table. Only during the first installation it is equal to the level of the application; then it will be increased every time there is a new PRIZES list. the “level” token of the first object.

4. To periodically update the TOUR GUIDE/2 “Special Offers” and “Prizes” working files, the EUROTRAVEL company prepares 2 change files with the following global names:

Adopted name	Considerations
EUROTRAVEL.TOURGUIDE2.SPECIALS.REF.<oldlevel+1>	The “level” token is obtained by incrementing the previous level.
EUROTRAVEL.TOURGUIDE2.SPECIALS.REF.<oldlevel+1>	The “level” token is obtained by incrementing the previous level.

5. When a fix to the Word Processor feature of TRAVEL OFFICE/2 R.3 application is available to be installed, the EUROTRAVEL company prepares a change file with the following global names:

Adopted name	Considerations
EUROTRAVEL.TRAV_OFFICE2.WORD_PROC. TEMPLATES.FIX.3.PROB0123	The fix affects only the “templates” of the Word Processor; last token is the EUROTRAVEL internally adopted problem identifier.

6. To install the upgrade of Word Processor for Travel Office/2 application, EUROTRAVEL uses a change file with the following Global Name:

Adopted name	Considerations
EUROTRAVEL.TRAV_OFFICE2.WORD_PROC.UPD.3.4	The change files contain only those file which have been changed in release 4 of the Word Processor feature; it does not refresh all the files in Word Processor release 3.

Note: At the end of the installation the FIX change file is no more tracked as installed; this reflects the real situation where the upgrade to next release had replaced the patched files.

7. To install the first OS/2 2.1 CSD, EUROTRAVEL prepares 4 function files:

Adopted name	Considerations
IBM.OS2V21_MAINT.SYSTEM.REF.1	The “component_id” token is not OS2V21 because EUROTRAVEL is considering the Maintenance System as being a new software.
IBM.OS2V21_MAINT_LAPS.REF.110	As this change file configure LAPS inside the Maintenance System, EUROTRAVEL decides that the second token must be the same with the previous change file.
IBM.OS2V21_MAINT.NVDM2_V20_CLT.REF.5	As this change file configure NetView DM/2 CC Clients inside the Maintenance System, EUROTRAVEL decides that the second token must be the same with the previous change file.
IBM.OS2V21.UPD.0.1	As the CSD updates the OS/2 2.1 files from maintenance level 0 to maintenance level 1, EUROTRAVEL decided to adopt the UPD global name.

Note: At the end of the installation, there are the Install Histories for the following Catalog entries:

IBM.OS2V21.REF.0
IBM.OS2V21.UPD.0.1

which reflects the real history of the OS/2 2.1 maintenance.

In the Catalog there are also the histories for the 3 change files that installed the Maintenance System. The CSD installation deleted that Maintenance System, but that action was not reflected into the Catalog. Therefore, it is necessary to manually delete the Install History if you want to have a Catalog tracking which is congruent with the workstation status.

8. To install the latest NetView DM/2 CSD, the EUROTRAVEL company applies the CSD on the NetView DM/2 diskette images, and prepares a change file with the following global name:

Adopted name	Considerations
IBM.NVDM2_CLIENT_V20.REF.6	As the CSD is not applied to the CC Client code but the CC Client code is reinstalled from the patched images, then EUROTRAVEL uses the REF global name, with “level” token reflecting the new NetView DM/2 Maintenance Level.

Final track of the Install History

After this maintenance activity, the SYSLEVEL command returns a partial status of the software installed on the target workstation. This is due to the lack of a standard for updating the SYSLEVEL on the work-station. Only the IBM applications and the fixes installed through CSDs do update the system; a few vendor applications use the SYSLEVEL files.

But the operator at CC Server can have the complete status by listing all the software that was installed on the controlled workstation and obtaining the following report.

Table 1-3. Global Names with Installation History	
Global Name	**Description**
IBM.OS2V21.REF.0	OS/2 2.1 code
IBM.OS2V21.UPD.0.1	1st CSD of OS/2 2.1
IBM.NNDM2_V20_CLIENT.REF.6	NetView DM/2 CC Client refreshed with 6th CSD
IBM.LAPS.REF.110	LAPS 1.1 code
EUROTRAVEL.TRAV_OFFICE2.MAIL.REF.3	Travel Office rel.3 Electronic Mail feRele
EUROTRAVEL.TRAV_OFFICE2.WORD_PROC.REF.3	
EUROTRAVEL.TRAV_OFFICE2.WORD_PROC.UPD.3.4	
EUROTRAVEL.TOURGUIDE2.APPLICATION.REF.1	
EUROTRAVEL.TOURGUIDE2.SPECIALS.REF.4	
EUROTRAVEL.TOURGUIDE2.PRIZES.REF.3	