

Migrations from Development or Maintenance

Applications or objects can be migrated into PAC from a development or maintenance status to Control, or to a test or production status. Migrations from development or maintenance may be processed in batch or online.

Saved Natural objects are compiled unless the dynamic source option is specified. Natural command processors are migrated "as is"; that is, they are not compiled during migration processing. The same applies to foreign datasets.

Not all objects will exist in catalog form; for example, copycode, text, macros, and recordings exist only in source form. Programs also may exist in source form if the PAC dynamic source option is in use.

For foreign objects, the existence of source and object code is predefined by the PAC administrator for each class of object.

Views/DDMs and rules which are required during the compile of Natural objects must already have been defined to PAC in a previous Predict migration event. Only then are they used for the compilation. They are treated as Natural objects and are identified with Natural object lists.

When migrating objects from development or maintenance status types, the Expand function is valid only if the cross-reference data exists for the object on the Predict file specified for the application status link. The EXP (Expand) command and special options of the SEL (Select) command use cross-reference (Xref) data from Predict for migrations from development or maintenance.

If the Copy option is specified for the migration, a copy of the objects remains in development or maintenance. If the Move option is specified, the objects are purged from the development or maintenance status.

Batch migrations from development (but not maintenance) can include input from an externally created work file by setting the Include option (instead of Copy or Move) in the migration path definition, or when the event is authorized. The work file can also be specified for CMWKF01 in the TRNEVENT step of the migration job. Only saved objects are processed and all objects are compiled as usual.

This chapter covers the following topics:

- Compiling Natural Objects in PAC
- Using the Rolling Facility

Compiling Natural Objects in PAC

PAC invokes the standard Natural CATAL utility to compile objects; thus, the compile process is the same when using PAC as when using the CATAL utility alone. The CATAL utility resides in library SYSLIB on the Natural system file FNAT.

All or None Compile

PAC considers a migration event to be a "unit of work" requiring that all or none of the objects in that unit be migrated. By default, then, PAC terminates a migration event if even one of the objects in the object list does not compile successfully. You can then correct the error and restart the migration. A restart causes all objects to be recompiled. Parameters, views/DDMs, and rules may also be reestablished.

User Exit 6 and Applymod 14 are available to customize the compile process.

User Exit 6

User Exit 6 can be used to override the "unit of work" default. It is invoked when a compile error occurs but before the migration event is terminated. It provides information about the total number of objects to be compiled and the total number of objects that did and did not compile. You can then decide whether to

- terminate the migration; then correct the error and restart the migration (the default); or
- migrate objects that compile now and then migrate other objects later after they have been corrected.

If you choose to migrate compiled objects now and uncompiled objects later, the source of the object(s) that did not compile is stored in PAC (Control status) under a new object version. If the Move option is specified, the object is purged from the origin status.

You can then migrate the uncompiled source to development or maintenance, correct the error, and migrate it back into PAC as another new version.

Applymod 14

If you use User Exit 6, you can then use Applymod 14 to ensure that an object that does not compile is not stored in PAC and a new version of the object is not created. Applymod 14 also ensures that the object will not be purged from the origin status, even if the Move option has been specified.

Natural Optimizer Compiler Error

If the MCG option is in effect, and the Natural Optimizer Compiler has not been linked to the Natural nucleus under which the migration is being processed, the object will still be compiled, but without the Natural Optimizer Compiler.

This error does not invoke User Exit 6 because the object does compile; however, the compiled code is not optimized.

Even though the object is processed normally, a NAT0893 error is returned and a warning message appears in the audit report of the event.

The MCG option can be overridden within the program itself; otherwise, the defaults specified on the PAC application definition are used.

View Security Restrictions Error

Compilation errors may occur due to violations of view security restrictions (for example, an attempt to update an "access only" file; or an attempt to read and/or update a "disallowed" file).

Using the Rolling Facility

Subordinate objects are those objects "used" when compiling another object. Subordinate objects include maps, copycode, data areas, rules, and views. By default, PAC uses the current versions of subordinate objects to compile new objects. However, because PAC represents applications in different phases of their life-cycles, it may be necessary to compile objects with specific earlier versions of subordinate objects.

The PAC rolling facility allows new versions of objects to be compiled with specific earlier versions of subordinate objects. Rolling can occur only when objects are migrated into PAC from a development or maintenance status since only these migrations result in the compilation of migrated objects.

The following subordinate objects may be rolled:

- The source form of copycode and maps.
- The cataloged form of data areas (local, parameter, and global).
- Views and rules.

If other types of objects are specified for rolling, PAC ignores them.

Specifying Objects to be Rolled

You can specify a subordinate object for rolling by attaching a reference to the object list entry. This reference instructs PAC to temporarily roll out the current version of the subordinate object and then roll in the older version of that same subordinate object.

When the migration is completed, the current version of the subordinate object is restored so that it may be used again as the default subordinate for any new compiles.

The reference you attach to an object list entry must indicate either a specific version number or a specific status:

- A specific version reference ensures that the correct object is rolled in, even if the object has been superseded. For example:

MAINGDA,G,0001

where the reference is a specific version number (0001).

- A status reference allows rolling to proceed without an exact version number. For example:

MAINCOPY,C,PRODUCTION

where the reference is to whatever version exists in a particular status (PRODUCTION).

While the version number method is a fixed definition and guarantees that the same version is always used, the status reference is dynamic; that is, PAC determines at processing time which version is at the specified status and then rolls in that version.

Since migration events are serialized for this purpose, the references are always guaranteed; this is important when versions of objects currently in a specific status are superseded. The reference method allows the same object list to be reused with subsequent migrations without the potential for loss of integrity.

Rolling Example

A typical example of the rolling procedure is presented below to demonstrate the need for and to further clarify the use of rolling.

1. The programmer develops Phase 1 of an application and migrates the objects into the PAC controlled environment. One of the objects is a global data area (GDA).
2. The application (including Version 0001 of the GDA) is implemented into production as required for use by the end user.
3. The programmer at a later point develops Phase 2 of an application and migrates the objects into the System-Test status in the PAC controlled environment where the project leader tests the new phase. At this point, the GDA has been changed and recompiled; that is, Version 0002 of the GDA is in the System-Test status, while Version 0001 of the GDA is in production.

4. The end user then encounters a bug in Phase 1 of the application in production and the programmer subsequently discovers that a program that uses the GDA must be changed.
5. The programmer changes the program and when he creates the object list, he ensures that the GDA is also included, except that the GDA entry in the object list specifies a reference (Production or 1 for Version 0001 of the GDA).

If the GDA reference is not specified in the object list, PAC recompiles the program with the current version of the GDA (Version 0002 in this case) and a NAT0933 error is received when the program is migrated into production.

6. PAC processes the object list as follows:
 - PAC recognizes the referenced GDA in the object list and rolls in that version of the GDA to the PCF system file (the work area for compiling).
 - The compile is performed; PAC recognizes that the production version (in this case, Version 0001) is to be used.
 - After the compile, the program is migrated into production once again and the NAT0933 error is not incurred because the program was compiled with the correct version of the GDA.
 - During the unlock step of the migration, the latest version of the GDA is then rolled back to the PCF system file, ready for the next compile that may use the latest version of the GDA by default.