

Using PAC

PAC controls the application life-cycle in the following ways:

- Secures the locations of libraries and system files
- Controls migrations and promotions of objects
- Protects source code
- Migrates code from a protected environment
- Ensures that object relationships are up to date
- Tracks maintenance activities

This section is organized in the following topics:

- Overview
 - PAC Entities
 - Migrations
 - Versioned Object
 - File Translation Table
 - Change Control Log
 - Maintenance Request
-

Overview

An application goes through a series of phases in its life-cycle. These phases typically include development, testing, implementation into production, and maintenance.

PAC allows you to monitor and control an application's life-cycle by defining a series of statuses through which it is promoted. Each status defines a phase in the application's life-cycle and the environment in which the application functions in that phase.

For example, an application might be brought into the PAC-controlled environment from a development status, move through several test statuses, and be implemented into a production status. When changes to the application are needed, the affected objects are copied to a maintenance status; after the changes are made, the objects might again move through one or more test statuses and back into a production status.

An application is promoted from one status to another through an online or batch migration. The migration process includes the following:

- defined origin and destination statuses,
- a list of the objects to be migrated,
- a scheduled migration event,

- job control statements (for batch migrations),
- an audit of the migration activities.

Controls and facilities are built into the migration process to ensure that the application is migrated to the correct location, the set of objects to be migrated is complete and correct, and the migration is made with the approval of the appropriate authority.

A logical migration is a promotion to the next status in the life-cycle; it may or may not include a physical migration of objects. A migration to a new physical location is both logical and physical; the logical and physical paths may differ. When two statuses share the same physical location, a migration from one status to the other is merely logical. Thus, all migrations have logical paths, but not all have physical paths.

Logical paths can be defined according to the needs of your site and applications. However, to help you maintain the integrity of applications at every stage of the life-cycle, the code is always physically migrated to and from the protected PAC ACF system file. The following figure shows how PAC controls the typical software life-cycle.

Status Type	Associated PAC Behavior
Development	Source code for objects can be modified freely. PAC recompiles each object migrated from a development status and assigns the object a new version number.
Test	Except for objects that use dynamic source variables, PAC migrates only executable code to a test status. Test statuses are unprotected by PAC; you can modify an application by migrating objects with Natural or PAC utilities. However, unless you use a PAC event to migrate the objects, PAC ignores them when it migrates the application to the next status; this restriction ensures the consistency of the application.
Production	All objects are protected by PAA and cannot be modified. Except for objects that use dynamic source variables, PAC migrates only executable code to a production status.
Maintenance	Source code can be modified freely. The PAC check-out/check-in facility tracks objects migrated to or from a maintenance status. PAC recompiles each object migrated from a maintenance status and assigns the object a new version number.

You can define multiple statuses for each basic type. For example, test statuses might include Systems Test, Integration Test, and User Test.

In addition to the basic types, PAC has the four special (PAC) status types shown in the table below. Source code for objects in these statuses cannot be changed.

Status Type	Associated PAC Behavior
Control	Files associated with the Control status contain the source and executable code for every version of every object under PAC control. Each application defined to PAC remains in the Control status even as the application moves through its life-cycle.
Archive	When an object is migrated to the Archive status, it is removed from all other statuses. The object can be unloaded to disk or tape and purged from the PAC system. Active production objects cannot be archived.
Incorporation	An incorporation status is used to bring existing production applications into PAC without recompiling the code.
Retire	A retire status is a logical or "virtual" status used to purge objects from a specified status (including the Control status). Inactive or obsolete objects, including historical objects that have been archived, are good candidates for a migration to a retire status.

Application Status Link

Before it can be migrated or promoted to a status, an application must be linked to the status. The application status link specifies the application, the status, and the physical location where the application's objects are stored at that status:

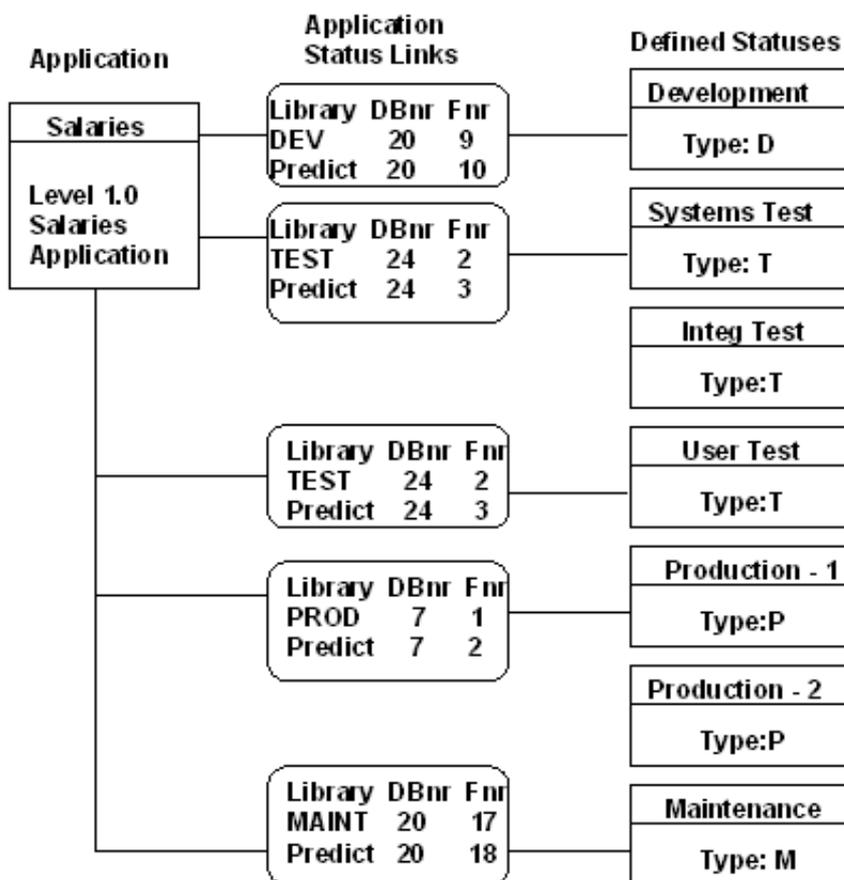
- The library and user system file for Natural programming objects,
- the dataset name and location for foreign objects,
- the Predict file for Xref data, rules, and views/DDMs.

The link definition may also specify a file translation table to be used in migrations to the status. The section File Translation Table later in this section describes the function of these tables.

For certain status types, the link definition may specify a step library to be used when PAC compiles objects during a migration to another status.

An application or object can exist in several statuses at the same time. Thus, an object in a production status could be copied to a maintenance status for changes while the original remains available to users.

The following figure shows an application linked to a subset of the defined statuses. It shows how each application status link defines the Natural library and Predict file for the application's objects in that status. Note that the links to Systems Test and User Test share a Natural library and Predict file.



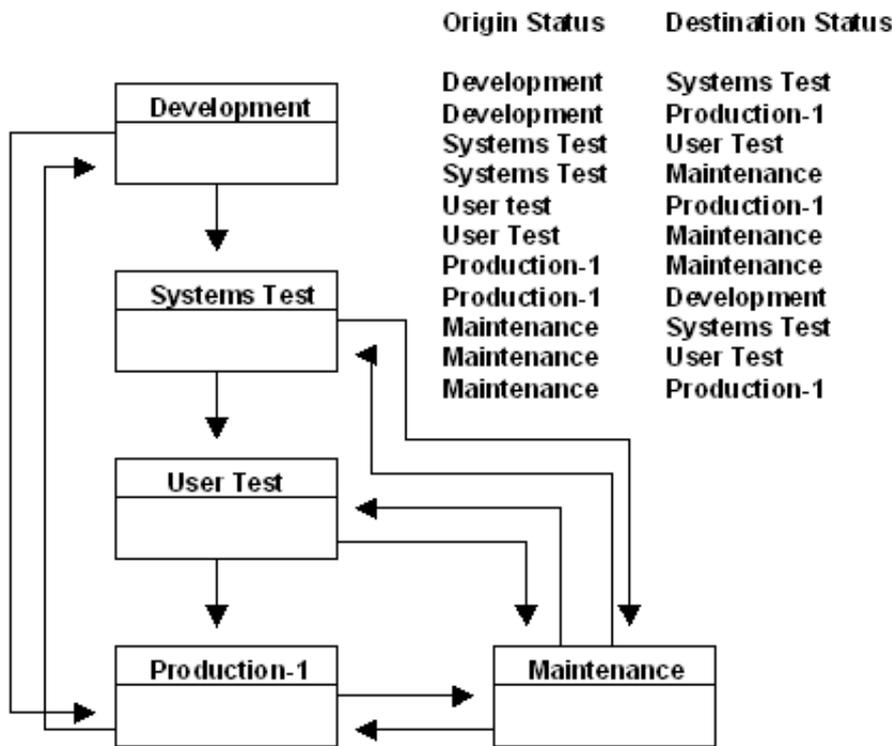
Migrations

As an application moves through its test plan or life-cycle, PAC uses defined procedures and controls to migrate it from one linked status to another. These procedures and controls are defined in the migration path and migration event.

Migration Path

A migration path specifies the origin and destination statuses of the migration. The origin status is the status from which the objects are migrated. The destination status is the status to which they are migrated.

A set of migration paths is application-specific. Before a migration path can be defined for an application, the application must be linked to the origin and destination statuses. The following figure shows migration paths defined for the application SALARIES.



Each migration path also defines the following defaults for migration events along that path:

- The migration mode:
 - Copy maintains the object version in the origin status and stores a copy in the destination status.
 - Move removes the object version from the origin status and stores it in the destination status.
 - Include copies objects from an external work file and stores them in the destination status (used when migrating objects into the PAC-controlled environment).
- The processing mode (batch or online).
- The Auto Expansion option, which specifies whether PAC automatically includes related objects in the list of objects to be migrated and how PAC identifies which versions of the objects to include.
- The PAC job, which contains job control statements for batch migrations.
- The Workfile Usage option, which specifies whether PAC migrates objects directly from one status to another or uses a work file in an intermediate step.

These defaults can be overridden when the migration event is authorized.

Migration Event

A migration event executes the migration. An event can migrate objects only along a defined migration path; this restriction controls the locations to which users can migrate objects.

When defining the migration event, you can select from a list of valid paths. The event can be scheduled for immediate or future processing, online or in batch. When authorizing a migration event, you can set system Applymods, which override PAC default processing procedures. For more information, refer to the section Applymods.

Object List

The migration event includes the object list of objects to be migrated, which may include an entire application or a single object. You can create the object list manually, or PAC can generate it automatically. PAC can also add related groups of objects automatically. See the sections "Expanding an Object List" and "Generating an Object List" for information about how PAC automates the task of creating object lists.

Enforcing Migration Procedures

PAC builds several levels of control into the migration path and migration event. Before a migration event can be processed, it must be authorized. Users who may authorize a migration along a path are specified in the migration-path definition; you can specify a single user as the authorizer, a range of users (for example, user IDs that begin with a specified set of characters), or a group ID to which individual user IDs are defined. You can also specify that the user who authorizes a migration event along the path must be different from the one who creates the event.

Similarly, an organization can restrict the authority to define migration paths. In a highly centralized organization, an administrator might be the only one authorized to define migration paths; thus, the administrator would define the migration paths for all applications at the site. In another organization, the project leader for each application might be authorized to define the migration paths for the application. By restricting the authority to define migration paths and authorize migration events, management can control migrations to specific environments.

Control Status and Migrations

Every application defined to PAC is automatically linked to the Control status, and every object migrated into PAC is migrated to Control. The Control status is associated with the protected PAC ACF system file, which stores the following:

- source and executable code for every object version in the PAC-controlled environment,
- object-version information,
- migration information.

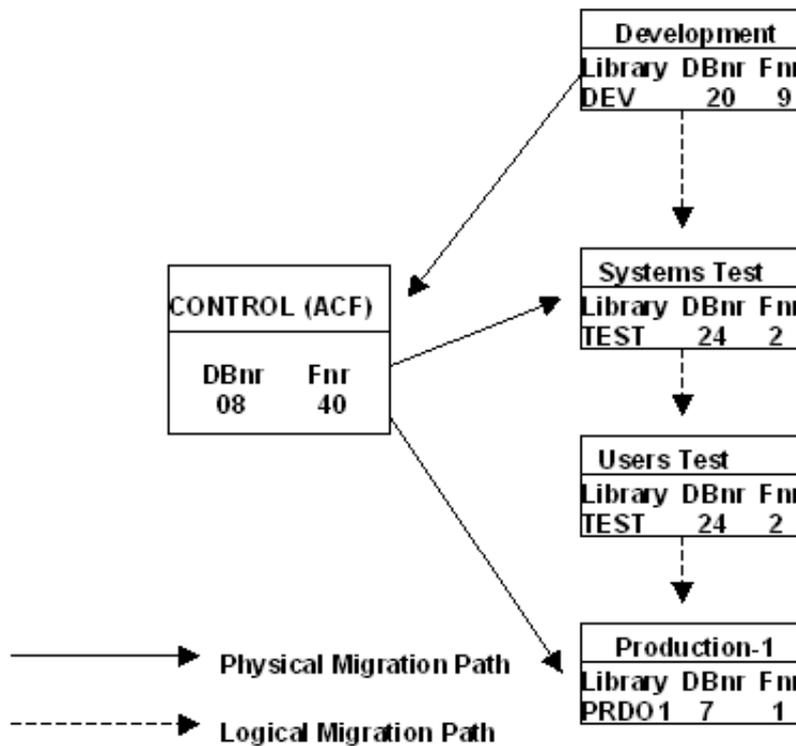
To ensure the integrity of applications at each stage of the life-cycle, the Control status and the ACF play a role in every PAC migration. When an object is migrated from a development or maintenance status, PAC compiles the object and stores the source and executable code in the ACF, along with information about the version and migration. If the destination status is a status other than Control, PAC then copies the executable object to the destination library.

Whenever a migration involves the physical movement of code, the objects are always migrated to the destination status from the ACF, even when Control is not defined as the origin status in the migration path. PAC copies the objects from the ACF into the library at the destination status and records the migration.

When a migration involves no change in physical location (that is, when the migration is a logical promotion only), PAC simply records the migration. The following figure shows logical and physical migrations of an application among five statuses to which it is linked.

The logical migration from Development to Systems Test involves an implicit migration to Control. Although Systems Test is the destination status in the migration event, the code is first compiled and stored in Control. Then it is copied to Systems Test from Control.

The Control Status and Migrations:



The table below describes the steps PAC performs to execute the migrations shown in the figure above:

Migration	Step	PAC Action
Development to Systems Test	1	Copies the objects from the library DEV.
	2	Compiles the objects and stores them, along with version information, in the ACF.
	3	Copies the executable objects from the ACF to the library TEST.
	4	Records the migration to Systems Test.
Systems Test to User Test	1	Records the migration to User Test.
User Test to Production-1	1	Copies the executable objects from the ACF to the library PROD1.
	2	Records the migration to Production-1.

An explicit migration to Control names Control as the destination status in the migration path. The Control status can be used as a staging area and a way to distribute authority. For example, when several programmers are responsible for different parts of an application, you might allow each programmer to authorize migrations to Control, but only the project leader to authorize a migration from Control to Systems Test. The programmers could independently migrate their parts of the application to Control. When all the parts have been compiled and stored in the ACF, the project leader could authorize an event migrating the entire application to Systems Test.

Versioned Object

PAC assigns a new version number to an object each time the object is migrated from a development or maintenance status. For a specific version of an object in a specific status, you can view the date and time it was saved and cataloged, the event that migrated it to the status, other statuses in which the version exists, and related objects.

The object audit history shows every version of the object, the statuses to which each version has been migrated, and the dates and times of the migrations. You can view the source code for an object version.

You can use a PAC utility to compare objects that are either PAC controlled or objects that are not under the control of PAC. Objects can be compared on an individual basis, as well as on a mass basis. Natural, DDM's as well as Foreign objects can be compared. Results of this compare utility can be printed off or even stored in a work file for later use. Full details of this utility can be found in the PAC documentation.

File Translation Table

A PAC file translation table (FTT) enables you to execute applications against different databases and files without changing or recompiling the source code.

Normally, when the source code for an object is compiled, it references database and file numbers in the development environment. When the object is migrated to a location where the user databases and files have different numbers, the source code must be changed to reference the new numbers and then be recompiled. This step not only requires time and computer resources but can also introduce inconsistencies into the code.

In PAC, an FTT is assigned to an application status link. When you migrate an object to the status, PAC copies the object from the ACF and dynamically recompiles it, substituting the correct database and file numbers. Since PAC copies the executable object from the ACF without removing it, neither the source code nor the executable code in the ACF is affected.

The following tables show excerpts from two File Translation Tables (FTTs) for an application (status names are added for clarification). Note that in both tables, the origin database number (DBnr) and file number (Fnr) are those of the Development status; the numbers are always translated from the numbers referenced in the compiled objects.

Origin Status	DBnr	Fnr	Destination Status	DBnr	Fnr
Development	20	9	Systems Test	24	2
	20	10		24	3
	20	11		24	4

Origin Status	DBnr	Fnr	Destination Status	DBnr	Fnr
Development	20	9	Systems Test	27	1
	20	10		27	2
	20	11		27	3

In the Development status, the source code for the application objects references file numbers 9, 10, and 11 in DBnr 20. When an object is migrated from Development to Systems Test, PAC compiles it using the same numbers and stores it in the ACF (as discussed in the section The Control Status and Migrations).

PAC then copies the executable object from the ACF and dynamically recompiles it, changing DBnr 20 and file numbers 9, 10, and 11 to DBnr 24 and file numbers 2, 3, and 4, respectively. PAC stores the recompiled object in the application library at the Systems Test status.

When the object is migrated from Systems Test to User Test, PAC again copies the executable object from the ACF; dynamically recompiles it to reference file numbers 1, 2, and 3 in DBnr 27; and stores the recompiled object in the application library at the User Test status. Although Systems Test is the origin status in the migration, PAC translates the DBnr and file numbers referenced in Development to those referenced in User Test.

You can specify that an FTT be available for all applications and statuses and either a Test or Production status,

, or you can restrict it to a specific application and/or status. By restricting an FTT, you prevent it from being applied to objects in unauthorized locations. For example, it might be necessary to prevent access to production data from anywhere but the production environment.

Change Control Log

The check-out/check-in facility is activated automatically by a migration to or from a maintenance status. When an event migrating an object to a maintenance status is processed, PAC creates a change control log for the check-out, which records the following information:

- Application name,
- Object name and version,
- User ID of the user checking it out,
- Terminal ID or batch ID,
- Date and time of the check-out,
- Library and user system file to which the object is migrated,
- Maintenance request ID (if applicable),
- Optional notes by the user.

PAC logs the same information, including the new version number, when the object is checked in from the maintenance status.

PAC audits check-out and check-in activities. A user exit lets you view the audit information, validate the user and object, and disallow the check-out or check-in. The user exit also lets you track and control concurrent maintenance activities. When a user checks an object out, you can list other users who currently have the same object checked out. When a user checks an object back in, you can list other users who still have the object checked out, or who checked the object in after this user checked it out. This mechanism currently only exists for Natural objects.

Maintenance Request

Maintenance requests are an optional PAC feature for documenting problems and maintenance activities. In addition to describing the problem and related activities, a maintenance request can list all objects required to resolve the problem; PAC can then generate an object list automatically from the list in the maintenance request. You can define and assign site-specific codes to indicate the state of a maintenance request (for example, open or closed) and the action to be taken in response. You can also prioritize requests.

When you assign a maintenance request to an event migrating an object to or from maintenance, PAC automatically assigns the request to the change control log. You can select and view change control logs related to the maintenance request.

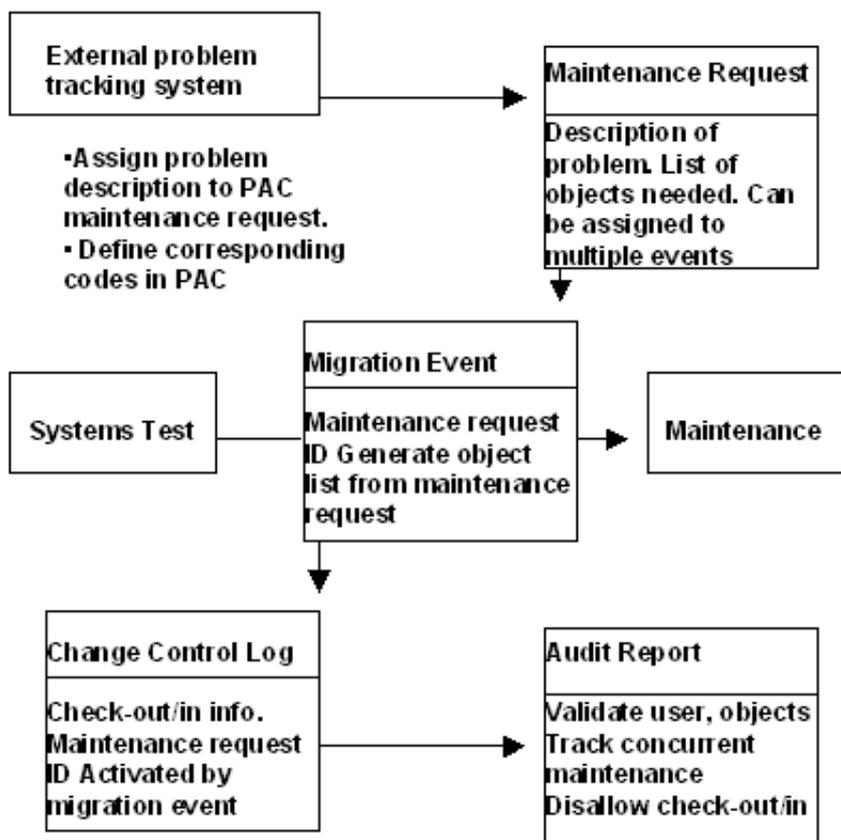
Maintenance requests can be used to integrate PAC with a user-developed problem-tracking system. You can define the PAC maintenance request ID to match the problem ID in the external problem-tracking system. The problem description in the external system can be assigned as an attribute to the PAC maintenance request. The codes for PAC problem states, priorities, and actions can be defined to correspond with codes in the external system.

As related problems emerge, they can be added to a request. You can use maintenance requests to group related problems in two ways:

By defining related problems in a single maintenance request and assigning the request ID to multiple migrations, you can view all migration events related to a problem or assign them to a group migration event. This concept is useful for grouping together all changes that made up a certain release level of an application.

To document complex problems and activities, you can group requests, each of which may be assigned to multiple migrations.

The following figure shows the relationships among PAC migrations to and from maintenance change control logs and audits, and maintenance requests.



With PAC user exits and application program interfaces, the system shown above can be tailored precisely to user requirements.

